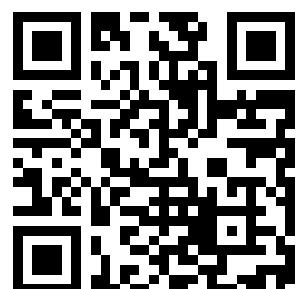

This is a reproduction of a library book that was digitized by Google as part of an ongoing effort to preserve the information in books and make it universally accessible.

Google™ books

<http://books.google.com>





Principles of Knowledge Representation and Reasoning

Proceedings of the
Second International
Conference

Cambridge, Massachusetts
April 22–25, 1991

Edited by:

James Allen
Richard Fikes
Erik Sandewall

With support from the American Association for Artificial Intelligence (AAAI), the International Joint Conference on Artificial Intelligence (IJCAI), and the Canadian Society for the Computational Studies of Intelligence (CSCSI).

Principles of Knowledge Representation and Reasoning

Proceedings of the Second International Conference (KR91)

THE MORGAN KAUFMANN SERIES
IN REPRESENTATION AND REASONING

Series editor, Ronald J. Brachman (AT&T Bell Laboratories)

BOOKS

James Allen, James Hendler, and Austin Tate, editors
Readings in Planning (1990)

James F. Allen, Henry Kautz, Richard Pelavin, and Josh Tenenberg
Reasoning About Plans (1991)

Ronald J. Brachman and Hector Levesque, editors
Readings in Knowledge Representation (1985)

Ernest Davis
Representations of Commonsense Knowledge (1990)

Matthew L. Ginsberg, editor
Readings in Nonmonotonic Reasoning (1987)

Judea Pearl
*Probabilistic Reasoning in Intelligent Systems:
Networks of Plausible Inference* (1988)

Glenn Shafer and Judea Pearl, editors
Readings in Uncertain Reasoning (1990)

John Sowa, editor
*Principles of Semantic Networks: Explorations in the
Representation of Knowledge* (1991)

Daniel S. Weld and Johan de Kleer, editors
Readings in Qualitative Reasoning about Physical Systems (1990)

David E. Wilkins
*Practical Planning:
Extending the Classical AI Planning Paradigm* (1988)

PROCEEDINGS

Principles of Knowledge Representation and Reasoning
Proceedings of the Second International Conference (KR91)
edited by James Allen, Richard Fikes, and Erik Sandewall

*Proceedings of the First International Conference on Principles of
Knowledge Representation and Reasoning (KR89)*
edited by Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter

*The Frame Problem in Artificial Intelligence:
Proceedings of the 1987 Conference*
edited by Frank M. Brown (1987)

*Reasoning about Actions and Plans:
Proceedings of the 1986 Workshop*
edited by Michael P. Georgeff and Amy L. Lansky (1987)

*Theoretical Aspects of Reasoning about Knowledge:
Proceedings of the First Conference (TARK 1986)*
edited by Joseph P. Halpern
Proceedings of the Second Conference (TARK 1988)
edited by Moshe Y. Vardi
Proceedings of the Third Conference (TARK 1990)
edited by Rohit Parikh

Principles of Knowledge Representation and Reasoning

Proceedings of the Second International Conference (KR91)

Edited by

James Allen
(University of Rochester)

Richard Fikes
(Stanford University)

Erik Sandewall
(Linkoeping University)

MORGAN KAUFMANN PUBLISHERS, INC.
SAN MATEO, CALIFORNIA

Editor *Michael B. Morgan*
Production Editor *Sharon Montooth*
Logo Design *Kathryn Finter*
Compositor *Technically Speaking Publications*

ADP 2956

Library of Congress Cataloging-in-Publication Data

Principles of knowledge representation and reasoning : proceedings of the second international conference / edited by James Allen, Richard Fikes, Erik Sandewall.

p. cm. -- (The Morgan Kaufmann series in representation and reasoning)

Includes bibliographical references and index.

ISBN 1-55860-165-1

1. Knowledge representation (Information theory)--Congresses.

2. Reasoning--Congresses. I. Allen, James. II. Fikes, Richard E. III. Sandewall, Erik. IV. Series.

Q387.P75 1991

003'.54--dc20

91-13453

CIP

ISBN 1-55860-165-1

MORGAN KAUFMANN PUBLISHERS, INC.
2929 Campus Drive, San Mateo, California 94403
(415) 578-9911
© 1991 by Morgan Kaufmann Publishers, Inc.
All rights reserved.
Printed in the United States of America

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the prior permission of the publisher.

95 94 93 92 91 5 4 3 2 1

Acknowledgements

KR91 would not have been possible without the extensive efforts of a great number of dedicated people.

First, and most important, is our outstanding program committee, who were asked to contribute extraordinary effort in reviewing and comparing close to 200 papers, and who did an excellent job:

Daniel Bobrow
Xerox PARC

Ronald Brachman
AT&T Bell Labs

Johan de Kleer
Xerox PARC

Jon Doyle
MIT

Didier Dubois
Universite Paul Sabatier

David Etherington
AT&T Bell Labs

Kenneth Forbus
Northwestern University

Dov Gabbay
Imperial College

Peter Gardenfors
Lund University

Hervé Gallaire
European Computer-Industrial Research Center

Mike Genesereth
Stanford University

Michael Georgeff
Australian AI Institute

Matt Ginsberg
Stanford University

Pat Hayes
MCC

David Israel
SRI International

Henry Kautz
AT&T Bell Labs

Kurt Konolige
SRI International

Robert Kowalski
Imperial College

Ben Kuipers
University of Texas

Hector Levesque
University of Toronto

Vladimir Lifshitz
University of Texas

David Makinson
Paris

Bill Mark
Lockheed AI Center

Dave McAllester
MIT

Ramesh Patil
Information Sciences Institute

Ray Reiter
University of Toronto

Len Schubert
University of Rochester

Yoav Shoham
Stanford University

Brian Smith
Xerox PARC

Austin Tate
AI Applications Institute

Robert Wilensky
University of California, Berkeley

We would like to especially thank program committee members David Israel and Hector Levesque for their extra efforts in providing reviews of papers whose initial reviews were missing or conflicting.

A special acknowledgement is due to Peter Gardenfors and Chuck Rich for their invited talks. The significant effort they put into their presentations is a major contribution to the conference.

We would also like to especially thank Jon Doyle, Bill Mark, and Bob Neches for organizing and moderating our three special sessions. Each spent a great deal of time and effort in planning his session, cajoling the participants into appearing, and making his session a stimulating and informative addition to the conference.

We would like to thank reviewers Anand Rao and Lynn Stein who assisted the program committee in their efforts to evaluate the papers.

Jim Schmolze did a tremendous amount of work doing the local arrangements, and much of the success of the actual conference is solely due to him. Thanks also to Caroline Miner, who managed all the actual detailed arrangements for the conference.

The support of Morgan Kaufmann in the production of these Proceedings was also a crucial ingredient. Thanks go to Mike Morgan, and especially Sharon Montooth for handling the production. Finally (but by no means least) we could not have survived without the patience and support of Judith Hook, Barbara Fikes, and Margareta Sandewall.

We would also like to acknowledge the generous support of the organizations that sponsored KR91: The Canadian Society for Computational Studies of Intelligence (CSCSI), the American Association for Artificial Intelligence (AAAI), and the committee for the International Joint Conference on Artificial Intelligence (IJCAI). We also wish to thank Linkoeping University, Price Waterhouse, and the University of Rochester for their cooperation.

Preface

The idea of explicit representations of knowledge that are manipulated by general-purpose inference algorithms dates back at least to the philosopher Leibniz. Leibniz envisioned a calculus of propositions that would exceed the differential calculus he had developed in its scope and power. But only very recently has it been possible to design automated systems that perform tasks partly by reasoning over a body of represented knowledge in this long-imagined way. And despite the success of many such "knowledge-based" systems in carefully delineated domains, core characteristics of intelligent behavior such as generality, flexibility, and adaptability are still very far from being realized. It is becoming increasingly clear that understanding precisely the nature and limitations of reasoning based on manipulation of explicitly represented knowledge is a central activity in the development of intelligent computer systems. The focus of this conference is research whose objective is to increase the understanding of the *principles* of representation and reasoning which govern knowledge-based systems.

The goal of the Knowledge Representation series of conferences is to bring together researchers interested in the principles of knowledge representation and reasoning on a regular basis. The enthusiastic success of the first conference led to this conference and its success leads us to believe that an ongoing series of biennial international conferences on the principles of representation and reasoning is warranted. KR conferences will provide an important venue for research results where the highest quality technical work that is sometimes too specialized for a general AI audience can be presented and discussed. Moreover, our goal is to encourage informal interactions by keeping the size of the conference small and avoiding extensive parallel sessions, and to provide lengthy and detailed papers in our Proceedings. These Proceedings represent the written record of KR91, the second conference in this ambitious series.

The response to the KR91 call for papers was extremely gratifying. We received close to 200 submissions. Because the conference is intended to be a small gathering of active researchers, this made the selection task extraordinarily difficult. Each submission was carefully reviewed by at least two members of our program committee. Because of our size constraints, the final selection of papers was very competitive. Some papers ended up being excluded not because there was anything specifically wrong with them, but simply because there were too many other submissions that appeared to be better. In the end, we managed to limit the program to the 55 outstanding papers contained in this volume. These papers represent, in our opinion, the very best work being done in the area of knowledge representation and reasoning.

In addition to presentations of the accepted papers, the conference program includes five special sessions featuring invited speakers and panelists who will present their views on some of the key issues of current knowledge representation and reasoning research. We trust that you will enjoy these provocative presentations and the discussions that have been planned. We are fortunate to have included here extended abstracts of the invited presentations and position papers of some of the panelists.

The ultimate success of this and future KR conferences depends on you, the people who contribute the papers, attend the conference, and read the proceedings. We thank you for your participation, and hope you find the efforts of the organizers and program committee rewarding.

James Allen
Conference Chair

Richard Fikes
Program Co-chair

Erik Sandewall
Program Co-chair

Contents

Acknowledgments	v
Preface	vii
Part I Submitted Papers	
Planning as Temporal Reasoning	3
<i>James F. Allen</i>	
Temporal Reasoning and Narrative Conventions	15
<i>Jonathan Amsterdam</i>	
Reflections about Reflection	22
<i>Giuseppe Attardi and Maria Simi</i>	
Nonmonotonic Inferences in Neural Networks	32
<i>Christian Balkenius and Peter Gärdenfors</i>	
Meta-Reasoning in Executable Temporal Logic	40
<i>Howard Barringer, Michael Fisher, Dov Gabbay, and Anthony Hunter</i>	
Pragmatic Logics	50
<i>John Bell</i>	
Metalogic for Knowledge Representation	61
<i>Antonio Brogi and Franco Turini</i>	
The Monotonic Abduction Problem: A Functional Characterization on the Edge of Tractability	70
<i>Tom Bylander</i>	
A New Formalisation of Subsumption in Frame-Based Representation Systems	78
<i>Jacqueline Castaing</i>	
Integrating Efficient Model-Learning and Problem-Solving Algorithms in Permutation Environments	89
<i>Prasad Chalasani, Oren Etzioni, and John Mount</i>	
Immortal: A Model-Based Belief Revision System	99
<i>Timothy S-C Chou and Marianne Winslett</i>	
Clustering Temporal Intervals To Generate Reference Hierarchies	111
<i>William S. Davis and James R. Carnes</i>	
Default Logic Revisited	118
<i>James P. Delgrande and W. Ken Jackson</i>	
Plan-Based Terminological Reasoning	128
<i>Premkumar T. Devanbu, and Diane J. Litman</i>	
A Formal Basis for Analogical Reasoning	139
<i>Charles Dierbach and Daniel L. Chester</i>	
The Complexity of Concept Languages	151
<i>Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt</i>	
Rational Belief Revision	163
<i>Jon Doyle</i>	
Conditional Objects and Non-Monontonic Reasoning	175
<i>Didier Dubois and Henri Prade</i>	

Completable Representations of Constraint Satisfaction Problems	186
<i>Eugene C. Freuder</i>	
A General Framework for Modal Deduction	196
<i>Alan M. Frisch and Richard B. Scherl</i>	
Credulous vs. Sceptical Semantics for Ordered Logic Programs	208
<i>D. Gabbay, E. Laenens, and D. Vermeir</i>	
Beyond Negation as Failure	218
<i>Hector Geffner</i>	
Disjunctive Defaults	230
<i>Michael Gelfond, Halina Przymusińska, Vladimir Lifschitz, and Mirosław Truszczyński</i>	
Partial Programs	238
<i>Michael R. Genesereth and Jane Yung-jen Hsu</i>	
Computational Considerations in Reasoning about Action	250
<i>Matthew L. Ginsberg</i>	
The Computational Value of Nonmonotonic Reasoning	262
<i>Matthew L. Ginsberg</i>	
Updates and Counterfactuals	269
<i>Gösta Grahne</i>	
Probably Approximately Optimal Derivation Strategies	277
<i>Russell Greiner and Pekka Orponen</i>	
Generalizing Prioritization	289
<i>Benjamin N. Grosof</i>	
Naming and Identity in a Multi-Agent Epistemic Logic	301
<i>Adam J. Grove and Joseph Y. Halpern</i>	
A Temporal Probability Logic for Representing Actions	313
<i>Peter Haddawy</i>	
Model Checking vs. Theorem Proving: A Manifesto	325
<i>Joseph Y. Halpern and Moshe Y. Vardi</i>	
Qualifying Number Restrictions in Concept Languages	335
<i>Bernhard Hollunder and Franz Baader</i>	
A Formal Theory of Spatial Reasoning	347
<i>Stephen G. Kaufman</i>	
S4 as the Conditional Logic of Nonmonotonicity	357
<i>Philippe Lamarre</i>	
On the Relation between Explicit and Implicit Belief	368
<i>Gerhard Lakemeyer</i>	
Toward a Metatheory of Action	376
<i>Vladimir Lifschitz</i>	
On the Difference between Updating a Knowledge Base and Revising It	387
<i>Hirofumi Katsuno and Alberto O. Mendelzon</i>	
Modal Nonmonotonic Logics: Ranges, Characterization, Computation	395
<i>Wiktor Marek, Grigori F. Shvarts, and Mirosław Truszczyński</i>	
Universal Attachment: An Integration Method for Logic Hybrids	405
<i>Karen L. Myers</i>	
Belief Revision and Default Reasoning: Syntax-Based Approaches	417
<i>Bernhard Nebel</i>	

Generalizing Atoms in Constraint Logic	429
<i>C. David Page, Jr. and Alan M. Frisch</i>	
A Theory of Inferred Causation	441
<i>Judea Pearl and T.S. Verma</i>	
The Logic of Epistemic Inconsistency	453
<i>Tarcisio Pequeno and Arthur Buchsbaum</i>	
The Utility of Consistency-Based Diagnostic Techniques	461
<i>Gregory M. Provan and David Poole</i>	
Modeling Rational Agents within a BDI-Architecture	473
<i>Anand S. Rao and Michael P. Georgeff</i>	
A Nonmonotonic Assumption-Based TMS Using Stable Bases	485
<i>William L. Rodi and Stephen G. Pimentel</i>	
Assertional Default Theories: A Semantical View	496
<i>Torsten Schaub</i>	
A Metalinguistic Treatment of Epistemic Contexts	507
<i>Paul Schweizer</i>	
The Completeness of DRAT, A Technique for Automatic Design of Satisfiability Procedures	514
<i>Jeffrey Van Baalen</i>	
The Feasibility of Defeat in Defeasible Reasoning	526
<i>Gerard Vreeswijk</i>	
Qualitative Intercausal Relations, or Explaining ``Explaining Away''	535
<i>Michael P. Wellman and Max Henrion</i>	
Qualitative Simulation with Multivariate Constraints	547
<i>Michael P. Wellman</i>	
The Ontology and Representation of Situations	558
<i>Robert Wilensky</i>	
Skeptical Reason Maintenance is Tractable	570
<i>Cees Witteveen</i>	
Part II Invited Presentations	
Nonmonotonic Inferences Based on Expectations: A Preliminary Report	585
<i>Peter Gärdenfors</i>	
Implemented Knowledge Representation and Reasoning Systems: An Endangered Species ?	591
<i>Charles Rich</i>	
Part III Position Papers from the Panel on Knowledge Sharing and Reuse	
Panel: Achieving Large Scale Knowledge Sharing	595
<i>William Mark</i>	
Knowledge Sharing: A Feasible Dream	597
<i>Giuseppe Attardi</i>	
Knowledge Interchange Format	599
<i>Michael R. Genesereth</i>	
The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases	601
<i>Thomas R. Gruber</i>	
Index	603
Author Index	609

Part I

Submitted Papers

Planning as Temporal Reasoning

James F Allen

Department of Computer Science

University of Rochester

Rochester, NY 14627

Abstract

This paper describes a reasoning system based on a temporal logic that can solve planning problems along the lines of traditional planning systems. Because it is cast as inference in a general representation, however, the ranges of problems that can be described is considerably greater than in traditional planning systems. In addition, other modes of plan reasoning, such as plan recognition or plan monitoring, can be formalized within the same framework.

1 INTRODUCTION

There is strong interest currently in designing planning systems that can reason about realistic worlds. In moving from the toy-world domains that characterized early work, researchers are looking at a wide range of issues, including reasoning in uncertain worlds, interacting with processes and events beyond the agent's direct control, and controlling mechanisms in real-time (i.e. robotics). One of the problems faced in extending existing frameworks is the weak expressiveness of the representation of the actions, events and the external world. This paper describes a reasoning system based on a temporal logic that can solve planning problems along the lines of traditional planning systems. Because it is cast as inference in a general representation, however, the ranges of problems that can be described is considerably greater than in traditional planning systems.

The key observations motivating this development are the following: 1) Actions take time - very few actions are instantaneous, 2) More than one action may occur at the same time, 3) Complex plans of activity may involve complex ordering constraints, and 4) Actions

may interact with external events beyond the agent's direct control.

Of these problems, the most central one is that of dealing with simultaneous action. Without simultaneous action, the range of problems that can be addressed is very limited, mainly restricted to specialized situations such as computer programming and game playing. It makes little sense to study planning in a dynamic changing world if the planner cannot act while some other natural event is occurring. The problems that arise when an action and a external event occur simultaneously exactly parallel the problems of simultaneous action. To understand why this problem is so difficult, it is important to look at the assumptions underlying the world representation in most current planning systems. This is examined in detail in the next section.

2 BACKGROUND: ACTIONS AS STATE CHANGE

The predominant approach to modeling action in artificial intelligence and computer science has been to view action as state change. This view underlies all the state-based planning systems (e.g. STRIPS and its successors), formal models of planning (e.g. the situation calculus [McCarthy & Hayes 1969], and work in dynamic logic for the semantics of programs (e.g. [Harel 1974] and [Rosenschein 1981]). In this view, the world is modelled by a succession of states, each state representing an instantaneous "snapshot" of the world. Actions are defined as functions from one state to another. Propositions in such models are relative to states. The notion of a proposition independent of a state is modelled as a function from states to truth values. In the situation calculus [McCarthy & Hayes 1969], such functions are called **fluent**s. For example, *On(A,B)* is a fluent that when applied to a state S is a proposition that is true if A is on B in state S.

In state-change models such as STRIPS, actions are instantaneous and there is no provision for asserting what is true while an action is in execution. Also, since the state descriptions do not include information about action occurrences, such systems cannot represent the situation where one action occurs while some other event or action is occurring. Finally, there is no reasoning about the future in these models except by searching through different possible action sequences.

Many non-linear planners suffer from the same deficiencies, although some (e.g. [Tate 1977, Vere 1983, & Wilkins 1988]) allow simultaneous actions if the two actions are independent of each other. In such cases, the effect of the two acts performed together is the simple union of the individual effects of the acts done in isolation. The problem with this solution is that it excludes common situations of interest in realistic domains. In particular, interesting cases of interaction occur when the effect of two actions done together is *different* from the sum of their individual effects. In particular, two actions may have additional effects when performed together, or they may partially interfere with each other's effects.

Here's one example concerning the door to the Computer Science Building at Rochester. The door is designed so that it requires both hands to open it, very annoying since you have to put down whatever you are carrying! There is a spring lock that must be held open with one hand, while the door is pulled open with the other hand. If we try to formalize this in a STRIPS-like system, we find there is no way to assert that unless the lock is held open it will snap shut.

An approach to this problem has been used in several systems (e.g. [Vere 1983]). The interaction of two actions is encoded in a special state. We'll call this technique **state encoding**. In particular, in the above example, we might introduce a fluent that is true only if the agent is holding the lock open. The action of holding the lock would be transformed into two actions, one to start holding the lock, and another to release it. Pulling the door simply has a precondition that the agent is holding the lock open. The fluent "holding lock open", once asserted by the TURN-LOCK action, remains true until a RELEASE-LOCK action deletes it. While this might solve this particular problem, there are many potential disadvantages with this approach. The first objection is that it is ad-hoc. While it may be the case that one can invent predicates to cover every specific example someone proposes, each must be done after the fact on an individual case by case basis. It is also not clear how the technique could be generalized to additional complexities involving simultaneous actions. More importantly, *holding the lock open* is intuitively an action - it may take effort on the part of the agent to maintain and must explicitly be part of the plan. This is not reflected in a representation where holding the lock

open is simply a fluent that will remain true until the agent does something to stop it.

In certain applications, where a detailed causal theory is known, state encoding approaches can be very powerful. If we cannot specify such a complete causal theory, however, or if we simply don't know enough about a situation to be able to use a causal theory, then other reasoning techniques must be explored. We would claim that both these problems arise in everyday planning situations: first, we have no detailed causal theory of the world, and second, we would not know the values for the parameters to the theory if such a theory was known.

2 TIMES, EVENTS AND ACTIONS

We do not have the space for an extensive discussion of time here. I will use interval temporal logic as developed in Allen [1983] and Allen and Hayes [1985]. In this logic, time is constructed out of one primitive object, the **time period**, and one primitive relation called **Meets**. Two periods meet if one precedes the other but there is no time between them. From the **Meets** primitive, many different temporal relations can be defined. In this paper uses the following:

- In*(*i,j*) - period *i* is contained in period *j*;
- Disjoint*(*i,j*) - *i* and *j* do not overlap in any way;
- Starts*(*i,j*) - period *i* is an initial subsegment of *j*;
- Finishes*(*i,j*) - *i* is a final subsegment of *j*.
- SameEnd*(*i,j*) - periods *i* and *j* end at the same time.
- Overlaps*(*i,j*) - *i* starts before but overlaps *j*

Time periods can be divided into the non-decomposable periods called **moments**, and decomposable periods, called **intervals**.

We attach times to predicates by adding an extra argument to each predicate as in Bacchus, Tenenberg, and Koomen [1989]. For example, the proposition *Green(FROG13,T1)* is true only if the object named by *FROG13* was green over the time named by *T1*.

By allowing time intervals as arguments we open the possibility that a proposition involving some predicate *P* might neither be true nor false over some interval *t*. In particular, consider a predicate *P* such that *p* is true during some subinterval of *t*, and also false in some other subinterval of *t*. In this case, there are two ways we might interpret the proposition $\neg P(t)$. In the **weak interpretation**, $\neg P(t)$ is true iff it is not the case that *P* is true throughout interval *t*, and thus $\neg P(t)$ is true if *P* changes truth values during *t*. In the **strong interpretation** of negation, $\neg P(t)$ is true iff $\neg P$ is true throughout *t*, and thus neither *P(t)* nor $\neg P(t)$ would be true in the above situation. Thus, a logic with only strong negation has truth gaps.

We use the weak interpretation of negation, as do Shoham [1987] and Bacchus, Tenenberg and Koomen

[1989], to preserve a simple two-valued logic. Of course, we can still make assertions equivalent to the strong negation. The fact that P is false throughout t can be expressed as follows, where In is true if its first temporal argument is contained in its second:

$$\forall t'. In(t', t) \supset \neg P(t').$$

This logic is still insufficient to conveniently capture many of the situations that we need to reason about, however. In particular, we need to introduce events as objects into the logic. Davidson [1967] argued that there are potentially unbounded qualifications that could be included in an event description. For example, the event of Jack lifting a particular ball might be asserted to occur at some time by a predicate *Lift*, as follows:

$$Lift(JACK34, BALL26, T1).$$

The problem arises in now representing the event "Jack lifted the ball onto the table". Either we need to add another argument to the *Lift* predicate, or we need to introduce a new predicate that represents a variant of lifting that includes an extra argument. Either is unsatisfactory. Davidson suggested the solution of reifying events, whereby additional modifiers would simply become additional predication on the event. Thus, the event of Jack lifting the ball to the table with the tongs might be represented as

$$\exists e . Lift(JACK34, BALL26, e, T1) \wedge \\ Dest(e) = TABLE555 \wedge \\ Instrument(e) = TONGS1.$$

We will represent knowledge about action in several ways. The first is by defining the necessary conditions for the event consisting of the action occurring (as in [Allen(1983)]) For example, consider the action *stack(a,b)*, which involves stacking block a on block b . For this action, we define a predication *Stack(a,b,e,t)*, that is true if e is an event consisting *stack(a,b)* occurring over time period t .

The event variable plays a central role - all the other parameters can be defined in terms of the event variable. For example, every instance of a stacking event uniquely defines the blocks that it involves, and the times relevant to the properties that define it. As a convention, we will denote the times for properties corresponding to preconditions by functions *pre1*, *pre2* and so on, those corresponding to effects by *eff1*, *eff2*, and so on, and those corresponding to conditions that hold while the event is occurring by *con1*, *con2*, and so on.

The stacking action in a typical STRIPS-style system is defined by its preconditions: (both blocks must be clear), and its transition function: (delete the formula *Clear(y)* and add the formula *On(x,y)*). We can use the STRIPS definition to motivate the conditions of the world that necessarily must be true whenever such a stacking event occurs.

In particular, each event type defines a set of temporal functions that define the structure of the temporal intervals involved in the event. For example, the class of stacking events uses functions to produce times corresponding to the properties involved in the action's preconditions and effects. We can define the structure of the stacking event as follows (see Figure 1):

Stacking Axiom 0: Temporal Structure

$$\forall e, \exists a, b, i . Stack(a, b, e, i) \supset \\ Overlaps(pre1(e), i) \wedge Finishes(con1(e), i) \wedge \\ Meets(pre1(e), con1(e)) \wedge Meets(i, eff1(e)) \wedge \\ SameEnd(i, pre2(e)) \wedge Meets(i, eff2(e)).$$

With this temporal structure defined for every stacking event, the axiom defining the necessary conditions for the event's occurrence now can be expressed as:

Stacking Axiom 1: Necessary Conditions

$$\forall i, a, b, e . Stack(a, b, e, i) \supset \\ Clear(a, pre1(e)) \wedge Holding(a, con1(e)) \wedge \\ Clear(a, eff1(e)) \wedge Clear(b, pre2(e)) \wedge \\ On(a, b, eff2(e)).$$

The above axiom asserts what is true whenever a stacking event occurs, independent of the situation. Other knowledge about action is relevant only in certain situations. For instance, if the block being moved in a stacking action was initially on another block, then this other block becomes clear (at least momentarily). This is easily expressed in the logic by the following axiom, which states that if block a was initially on another block c , then c becomes clear when a is moved:

Stacking Axiom 2: Conditional Effects

$$\forall i, a, b, c, t, e . Stack(a, b, e, i) \wedge On(a, c, t) \wedge Overlaps(t, i) \supset \\ Clear(c, eff3(e)) \wedge Meets(t, eff3(e)) \wedge \\ Meets(t, con1(e)).$$

This axiom applies in a situation with three blocks, say A , B and C where A is originally on block C . The conditions for the action *Stack(A, B, E1, T1)* are shown graphically in Figure 1. Note that this definition does not assert that the block C will be clear at the end of the stacking event. In particular, if two stacking events overlap in time (say *Stack(A,B,E1,T1)* and *Stack(D,C,E2,T2)*) then this may not be the case, for D may be placed onto C before A is placed on B . Such subtleties cannot be represented easily in a STRIPS-style representation.

The development so far has not captured any sense of causality. In particular, the axioms above do *not* state what properties are caused by the stacking action, or what properties simply must be true whenever the action succeeds. This is the distinction that STRIPS makes between preconditions and effects. Intuitively, it is clear that the stacking action causes block a to be on block b in situations where both blocks are clear at the start of the action. Furthermore, the stacking action

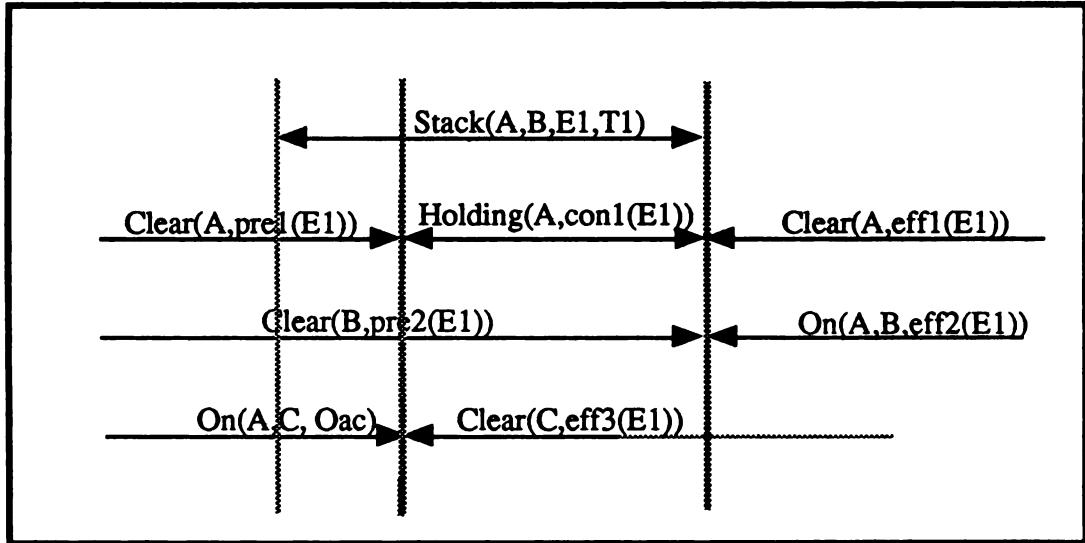


Figure 1: The necessary conditions for $\text{Stack}(A,B,E1,T1)$ in a situation where A is originally on C (using stacking axioms 1 and 2)

causes block b to become not clear while it doesn't affect the condition that block a is clear.

To encode such knowledge, we need to be able to reason about action attempts (cf. [McDermott 1986]). The logic developed so far can express the fact that a certain event occurred, but not that an agent attempted to do some action. The predicate Try is defined such that $\text{Try}(a,e,t)$ is true whenever the action a is attempted by the agent at time t in order that event e occurs. Of course, if the conditions are not right then the action will not succeed and the event does not occur. For example, $\text{Try}(\text{stack}(a,b),e,t)$ does not necessarily imply $\text{Stack}(a,b,e,t)$. The relationship between the two is defined by axioms corresponding to precondition assertions: In particular, we can assert that wherever the agent tries to stack a on b starting from an initial situation where a and b are clear, then a stacking event occurs:

Stacking Axiom 3: Prerequisites

$$\forall i,j,k, a,b,e . \text{Try}(\text{stack}(a,b),e,i) \wedge \text{Clear}(a,j) \wedge \text{Overlaps}(j,i) \wedge \text{Clear}(b,k) \wedge \text{SameEnd}(i,k) \supset \text{Stack}(a,b,e,i) \wedge \text{pre1}(e)=j \wedge \text{pre2}(e)=k.$$

3 THE PLANNING FORMALISM

A planning system can now be specified using the temporal logic developed above. This system can reason about certain classes of interacting simultaneous events and it has a limited capability for reasoning about the future. In particular, while it cannot plan to change any external events predicted to occur in the future, it can construct plans that take future events into account and reason about interactions with such events. Pelavin [1988] and Allen et al [1991] present an extended logic that can represent future possibilities as well.

In order to construct plans, an agent needs to predict future states of the world. STRIPS-like problem solvers do this by using the add and delete lists to transform the current state into the next state. With a representation based on an explicit temporal logic, however, it is more complicated. In particular, if a proposition P is asserted to hold at a time $T1$ and then some action A occurs after $T1$ that makes P false, it is still true that P held at time $T1$. So the representation of the world should still contain this assertion. What has changed once the new action is introduced is some prediction about whether P holds in the future. For example, before A is known about, the agent might have predicted that P still holds in the future. Once the action A is expected, however, this prediction might change.

Thus it is the predictions (or expectations) about the future that change as an agent plans. Since an agent may change its mind about what future actions it might do, most conclusions about the future must be retractable. This suggests that some form of non-monotonic reasoning is necessary in order to maintain the world model and some models such as deKleer's ATMS [deKleer 1986] might be useful. But there is a simpler route: a model can be outlined that views all predictions about the future as conditional statements based on what the agent assumes about the future including its own actions. Given an initial world description W and a goal statement G , the plan is a set of assumptions A_1, \dots, A_n such that

$$W \vdash (A_1 \wedge A_2 \wedge \dots \wedge A_n \supset G.)$$

Of course, if the A_i 's are inconsistent then this statement is vacuously true, so we must also add the condition that

$$A_1 \wedge A_2 \wedge \dots \wedge A_n \text{ is consistent.}$$

Finally, we want to avoid assuming the problem away. For instance, if A_1 is simply equivalent to the goal statement G , the above conditions are true but we can't say we've solved the problem! This is handled by restricting the form of assumptions that the planner can make, as described below.

With this analysis, we can view the planning problem as consisting of two types of reasoning:

- prediction - what is the world like (based on a given set of assumptions)
- planning - what assumptions should the agent make about its future behavior, and the future world.

These two types of reasoning are explored in detail in the remainder of this section.

3.1 Predicting the Future

If an agent had full knowledge about a world, then predicting the future would be a relatively well-defined task. The agent would simply simulate the future course of events starting from the present state. In practice, however, the agent never has such detailed knowledge about the world - the agent's knowledge of the present state is partial, and the world is not well-understood enough to make precise predictions. Even qualitative models, such as those discussed in Bobrow [1985], assume a level of knowledge about the state of the world and the processes that change it that are not realizable in most situations.

Here we develop a very conservative model of prediction based on maintaining limited consistency of the agent's beliefs about the world. Essentially, given some set of beliefs about the future, the predictions are simply what is inferable from those beliefs using the agent's knowledge of the structure of the world and the definitions of actions. The particular system we will specify uses a forward chaining strategy on Horn clauses coupled with constraint propagation techniques for time (Allen, 1983, Koomen, 1989) to make the predictions.

To drive the predictor, we need knowledge about the actions, such as defined in the last section, as well as general knowledge of the domain. For instance, to reason about the door latch problem given at the start of this paper, we would have to know that a door cannot be open and shut at the same time. This motivates a forward chaining rule that guarantees that this cannot occur:

Domain Constraint 1

$$\forall t1,t2 . \text{DoorOpen}(t1) \wedge \text{DoorClosed}(t2) \supset \\ \text{Disjoint}(t1,t2).$$

Similarly, the latch is either open or shut, but not both:

Domain Constraint 2

$$\forall t1,t2 . \text{LatchShut}(t1) \wedge \text{LatchOpen}(t2) \supset \\ \text{Disjoint}(t1,t2).$$

Allen & Koomen (1983) show that a prediction algorithm using action definitions and domain constraints similar to those presented here can capture many important aspects of non-linear planning systems.

3.2 Making Assumptions

There are two main classes of assumptions that the planner must make. It must decide what actions it will attempt to perform, and it must make assumptions about how the external world will behave. While the planner may assume it can attempt any action at any time, the action will only succeed if the appropriate conditions hold. As we'll see below, this is implemented by allowing the planner to add an assertion of the form $\text{Try}(a,e,t)$ without proof.

The assumptions about the external world are limited at present to *persistence* assumptions [Dean & McDermott 1987], that once a property is established, it tends to remain true until explicitly changed. More precisely, a literal $P(i_1, \dots, i_n, t)$ can be proven by persistence if there is a literal in the database of form $P(i_1, \dots, i_n, t')$ where it is possible that $t=t'$. This definition not only allows persistence into the future, it also allows persistence into the past. It will be examined further after the basic planning algorithm is presented.. Note that we have a constant-time method of heuristically checking whether $t=t'$ is possible given the network representation of temporal information: the system simply checks if " $=$ " is one of the disjuncts still present on the arc connecting the node for t and the node for t' (see [Allen 1983]).

4 THE PLANNING SYSTEM

To a first approximation, the planning algorithm is simply a backwards-chaining proof strategy driven by the goal statement, where assumptions about action attempts and persistence can be made without further proof. While very simple, this planner is similar in power to the regression planners (e.g. Waldinger, 1977).

To distinguish the logic developed so far from the system, which involves heuristic reasoning on a restricted formalism, we will use a different notation. A literal consists of a predicate name and a list of arguments enclosed in square brackets. Thus the literal corresponding to the formula $\text{On}(A,B,G)$ is:

[On A B G].

Knowledge about actions is captured by a set of planning rules, which are a modified notation of Horn clauses using "?" as a prefix to indicate variables. In general, a planning rule is of the form

$C << D1 D2 \dots Dn \text{ such that } A1, \dots, Ak$

and can be interpreted formally as a Horn clause: the consequent literal C is true (or provable) if the antecedent literals D₁,...,D_n are true and the constraints A₁,...,A_k are true. The system, however, will treat the antecedents and constraints differently to produce an efficient inference strategy.

Two planning rules for the action of pulling the door open are as follows: First, pulling on door when the latch is open results in the door being open:

(PullOpen.1)
 $[DoorOpen ?t'] << [PullOpen ?e ?t] \text{ such that } [EQ eff1(?e) ?t']$.

Second, you may open the door any time you try to, if it's closed and the latch is unlocked:

(PullOpen.2)
 $[PullOpen ?e ?t] << [DoorClosed pre2(?e)]$
 $[LatchOpen pre1(?e)]$
 $[Try [pull] ?e ?t]$

The temporal structure for the *PullOpen* action must also be defined in the system. Rather than present details we will summarize such information by an axiom in the original logic that defines two precondition intervals (for the latch being open, and the door being shut), and one effect interval (for the door being open):

PullOpen Axiom 0: Temporal Structure
 $\forall e, t. PullOpen(e, t) \supset OverlapsDuring(pre1(e), t) \wedge$
 $Meets(t, pre2(e)) \wedge Starts(eff1(e), t)$

Planning rules will be used by the system in both a forwards (i.e. from antecedent to consequent) and a backwards (from consequent to antecedent) chaining manner. To apply a rule in a backwards chaining manner, the rule's consequent C is unified with the goal. Then the constraints A₁,...,A_n are added to the database and the antecedent literals D₁,...,D_n are introduced as subgoals. To apply the rule in a forward manner, if a literal is added to the database that unifies with some antecedent literal D_i, and all the other D_j ($j \neq i$) and the constraints A₁,...,A_n are in the database, then we also assert the consequent C. For instance, rule (PullOpen.1) could be used to suggest that a goal [DoorOpen D01] could be accomplished by an event E1 if we can prove [PullOpen E1 T1] under the constraint eff1(E1)=D01. The same rule is also used to predict the consequence of the same event E1 occurring at time T1: i.e., if [PullOpen E1 T1] is added then add [DoorOpen eff1(E1)].

This simple example illustrates the basic technique for generating plans - planning rules are used to backward chain to suggested actions, and then in a forward manner to compute the consequences of those actions. In addition, all the domain prediction rules are also used in a forward chaining manner to compute additional

consequences of the action. For example, Domain constraint 1 above would be asserted as the following forward chaining rule:

$[Disjoint ?t1 ?t2] <\text{forward}>$
 $[DoorOpen ?t1] [DoorClosed ?t2]$.

All the other domain constraints can be expressed similarly.

There are several additional issues to consider before the final algorithm is specified. First, the planner must be able to create event structures as needed, since the appropriate events will not generally be known to occur in advance of the planning. This is accomplished during the backwards chaining phase: whenever a literal containing an unbound event variable is to be introduced as a goal, a new event constant is generated and the temporal structure associated with that event is added to the database together with any other constraints specified in the planning rule. As an example, given the goal [DoorOpen D01], rule (PullOpen.1) suggests a subgoal of proving [PullOpen ?e ?t]. Before this is considered further, a new event, say E1, and a new time, say T1, are created and the following constraints are added to the database from the definition of the temporal structure of Stack events (Stacking Axiom 0):

$[OverlapsDuring pre1(E1) T1]$
 $[Meets T1, pre2(E1)]$
 $[Start eff1(E1) T1]$.

What we have done is create the temporal structure for an event that could accomplish the goal clause. We have not asserted that this event yet occurs. This will require further chaining to prove [PullOpen E1 T1]. This process of creating event and temporal constants to replace the unbound variables will be called *instantiating* the planning rule.

4.1 The Algorithm

The following algorithm defines a planner that does not commit to the persistence assumptions until the entire plan is otherwise complete. It uses the simple backwards chaining technique from the goals as described informally above, and forward chaining to compute the consequences of its assumptions about persistence and action attempts. Because the temporal aspects of the plan are independent of the planning algorithm, the simple back-chaining strategy does not restrict the plans that can be produced.

It consists of two main parts: the **plan generator**, which creates a particular plan and the **assumption verifier**, which takes a suggested plan and evaluates whether the persistence assumptions that support it still appear to be consistent. Let GS be the goal stack, which is initialized to the initial set of goals when the algorithm is invoked. The output of this algorithm is a set of actions to be attempted by the agent (the **action list**), a set of assumptions about the world (the

assumption list), and the world state generated by the prediction reasoner. Each persistence assumption consists of a literal P true over some time period T and an equality relation involving T that captures the persistence assumption.

Plan Generation

This is a non-deterministic version of the planning algorithm. A PROLOG-style search strategy to iterate through all possible proof paths can be added in the obvious way.

(0) Do until GS is empty; then go to verification stage: remove the top element of GS and call it G;

(1) Choose:

(1.1) If a formula unifying with G is found in the database, then bind any variables in G as necessary;

(1.2) If G can be proven by a persistence assumption, then pass G to the prediction reasoner, and add G together with the equality assertion that justifies the assumption to the assumption list;

(1.3) If G is of the form $\text{Try}(A, e, t)$, for some action A, then add G to the action list, pass G to the prediction reasoner;

(1.4) Find a planning rule R whose consequent unifies with G, instantiate the rule as defined above (i.e. binding the event and temporal variables and adding the constraints) and push the antecedents onto GS.

Verifying Assumptions

This algorithm uses the temporal reasoner to check that all the persistence assumptions appear to be globally consistent. It does this by first re-checking the temporal constraints for each assumption individually to see if it is still possible. It then adds all the assumptions together to see if they appear to be globally consistent (according to the temporal reasoning algorithm). If some assumptions are no longer consistent, the planning stage is re-activated.

(2.1) Check each persistence assumption individually to see if it is still possible given the current temporal network generated by the prediction reasoner. If not, add the literal associated with each assumption that is now impossible to GS and restart at step (0).

(2.2) (Given that step (5) succeeded) Add the persistence assumptions to the prediction reasoner.

Unless the prediction reasoner returns an inconsistency, we are done. If an inconsistency is found, then we must select an assumption to retract. Designing a good strategy for this is left as future work. For now we simply select an assumption at random. Remember that assumptions consist of a literal P, and an equality assertion $t=t'$. Given the selected assumption, add $t \neq t'$ to the prediction reasoner, add P to GS and restart at step (0).

5 THE DOOR-LATCH PROBLEM

One of the major goals of this work was allowing plans that necessarily required simultaneous actions. The Door-Latch Problem was posed as the simplest example of this type of situation. In this section, we show how the domain can be formalized and a plan constructed from first principles that will open the door. First we present the planning axioms that define the actions, and then give an overview of the solution.

Remember that the complication in this example is that the agent must realize that it must continue to hold the latch open while it is pulling on the door. The actions needed are turning the latch, holding the latch open, and pulling the door. Unless the latch is held open, it snaps shut. Given that the planner uses persistence assumptions about the world, some technique must be introduced to prevent the planner from using this technique to infer that the latch stays open. This would best be handled by adding some causal reasoning to the predictor, but a simpler technique can be used in this class of situations. We will define the turn-latch action such that its effect holds exactly for a moment, i.e. a non-decomposable period. Thus any action that requires the latch to be open for an extended period of time cannot accomplish this by persistence, since a moment cannot be equal to an interval by definition. The hold-latch action is then defined so that it requires the latch to be open at the time the action starts (which may be a moment) and has the effect that the latch stays open for the duration of the action. Specifically, we have the planning rules below which are used by the predictor to maintain the world representation.

The temporal structures for each event are axiomatized below and shown graphically in Figure 2. The *PullOpen* action was defined earlier. *TurnLatch* events have a precondition interval (for the latch being closed) and an effect moment (for the latch being open):

TurnLatch Axiom 0: Temporal Structure

$\forall e, t. \text{TurnLatch}(e, t) \supset \text{Finishes}(t, \text{prel}(e)) \wedge \text{Moment}(\text{effl}(e)) \wedge \text{Meets}(t, \text{effl}(e)).$

HoldingLatch events define a single precondition period (for the latch being open) and an effect interval simultaneous with the event interval:

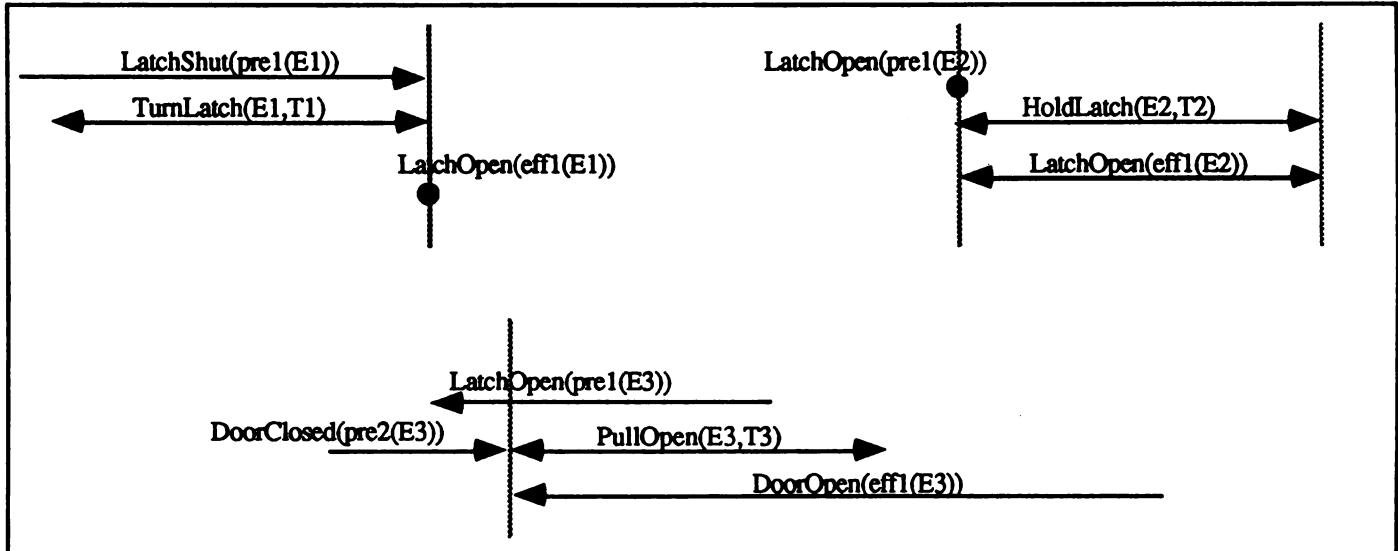


Figure 2: The temporal structure for three events in the door problem

HoldingLatch Axiom 0: Temporal Structure
 $\forall e, t. \text{HoldOpen}(e, t) \supset \text{Meets}(\text{pre1}(e), t) \wedge \text{EQ}(\text{eff1}(e), t)$

The planning rules for these actions are as follows:
 Turning the Latch has the effect that the latch is momentarily open:

(TurnLatch.1)
 $[\text{LatchOpen } ?t] <<$
 $[\text{Moment } ?t']$
 $[\text{TurnLatch } ?e ?t]$
 such that $[\text{EQ } ?t' \text{ eff1} (?e)]$.

Turning the latch can be accomplished by trying to do it when the latch is shut:

(TurnLatch.2)
 $[\text{TurnLatch } ?e ?t] <<$
 $[\text{LatchShut pre1} (?e)]$
 $[\text{Try [turnlatch] } ?e ?t]$.

The latch remains open if and only if it is held open. In particular, note that the effect and the action in this rule must be simultaneous:

(HoldOpen.1)
 $[\text{LatchOpen } ?t] <<$
 $[\text{Interval } ?t']$

$[\text{HoldOpen } ?e ?t]$
 such that $[\text{EQ eff1} (?e) ?t']$.

Holding the latch open succeeds whenever the latch is open at the start of the holding act:

(HoldOpen.2)
 $[\text{HoldOpen } ?e ?t] << [\text{LatchOpen pre1} (?e)]$
 $[\text{Try [holdopen] } ?e ?t]$

Assuming a situation, in which the agent is near the door, the initial world description would be as follows, where I is the current time, and G is the time when the goal must hold:

$[\text{In I ls1}]$	$[\text{LatchShut ls1}]$
$[\text{In I dc1}]$	$[\text{DoorClosed dc1}]$
$[\text{Before I G}]$	

The goal is simply to have the door open over time G, i.e. $[\text{DoorOpen do1}]$ such that $[\text{In G do1}]$. The initial planning situation is shown in Figure 3.

Here's a brief sketch of the planner in operation. Given the goal $[\text{DoorOpen do1}]$, rule (PullOpen.1) applies and introduces the subgoals after instantiation:

GS: $[\text{PullOpen E1 T1}]$
 where $[\text{EQ eff1(E1) do1}]$ and the temporal

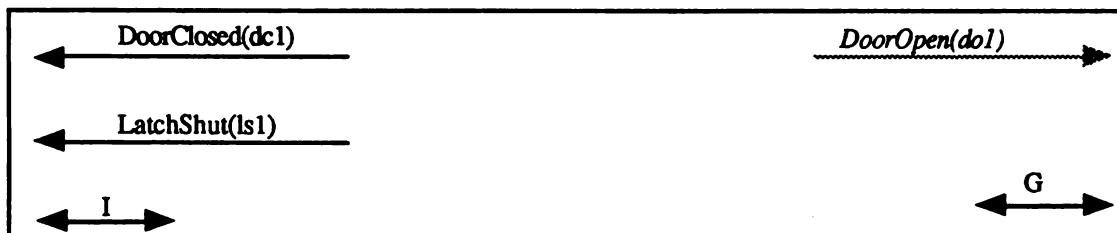


Figure 3: The Door-latch problem

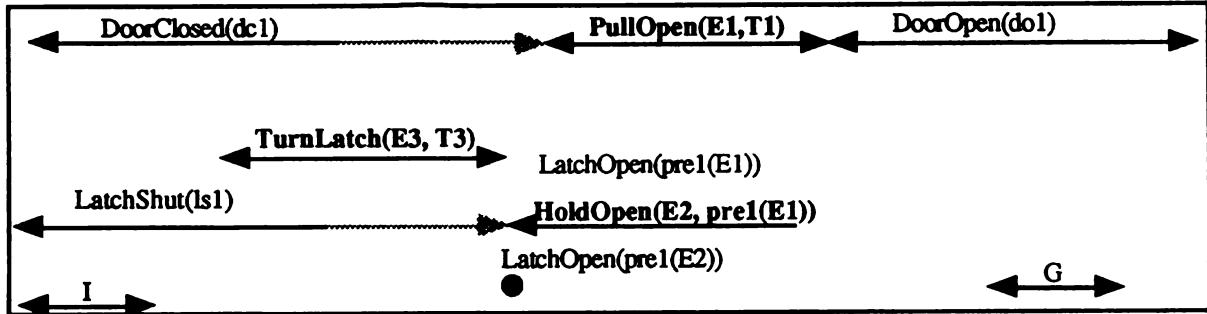


Figure 4: The solution to the Door-Latch Problem

constraints for the new PullOpen event E1, i.e. [OverlapsDuring pre1(E1) T1], [Meets T1 pre2(E1)] and [Starts eff1(E1) T1] are added to the database

The subgoal [PullOpen E1 T1] can be proven by rule (PullOpen.2) producing the new set of subgoals:

GS: [DoorClosed pre2(E1)]
 [LatchOpen pre1(E1)]
 [Try [pull] E1 T1].

The first subgoal is proven by persistence from the initial state, using the assumption that pre2(E1)=dc1. The second subgoal, [LatchOpen pre1(E1)], requires further planning. Rule (TurnLatch.1) cannot apply here as it requires the interval pre1(E1) to be a moment. Rule (HoldOpen.1) does apply, however, and introduces the subgoal [HoldOpen E2 pre1(E1)] (note that by the planning rule pre1(E1)=eff1(E2), which in turn equals the time of the HoldOpen event by its definition). Rule (HoldOpen.2) then applies to this subgoal and introduces the following subgoals after instantiation:

GS:[LatchOpen pre1(E2)]
 [Try [holdopen] E2 pre1(E1)]
 [Try [pull] E1 T1].

This time, rule (TurnLatch.1) can apply (since pre1(E2) can be a moment) and the action [TurnLatch E3 T3] is introduced. After using rule (TurnLatch.2) to reduce this goal, the following subgoals remain:

GS: [LatchShut pre1(E3)] [Try [turnlatch] E3 T3]
 [Try [holdopen] E2 pre1(E1)]
 [Try [pull] E1 T1].

The first of these subgoals can be proven by persistence, since it is possible that pre1(E3)=ls1, and the remaining three subgoals are trivially proven as they are under the control of the planner. As each of these is assumed, it is added to the database triggering the forward-chaining prediction rules. As a result, the door is predicted to be open at time do1.

Finally, the persistence assumptions must be verified, and then added to the predictor producing the final plan as shown in Figure 4, with the persistence assumptions

shown in grey. Note that the Pull action must start within the time when the HoldOpen action occurs, as desired. If this were not the case, the final effect, namely that the door is open, would not be predicted by the prediction mechanism. Thus we've shown that the planner can find the correct plan, and that it would not accept the faulty plan that would arise from a STRIPS-style planner, or from a naive persistence mechanism.

6. PLANNING WITH EXTERNAL EVENTS

Another simple example of some interest shows that the planner can co-ordinate with external events that it knows will occur sometime in the future. For example, consider a different initial situation, which is the same as before except that the planner knows that the door is unlocked automatically between 8AM and 9PM every day, and the goal is to get the door open sometime between 7AM and 9AM. This situation is shown in Figure 5, where the times of day are represented by moments.

The initial database consists of the following assertions:

[In I ls1]	[Meets ls1 lo1]
[LatchShut ls1]	[Meets ls1 8AM]
[LatchOpen lo1]	[DoorClosed dc1]
[In I dc1]	[Before I 7AM]
	[Before 7AM 8AM]
	[Before 8AM 9AM]
	[Before 7AM G]
	[Moment 7AM]
	[Moment 8AM]
	[Moment 9AM].

The initial goal is as before, to accomplish [DoorOpen do1] such that [In G do1]. Using rule (PullOpen.1) we get the subgoal of [PullOpen E1 T1], where eff1(E1)=do1. Rule (PullOpen.2) gives two preconditions for this action, namely

[DoorClosed pre2(E1)]
 [LatchOpen pre1(E1)].

In this case, both can be proven by persistence. [DoorClosed pre2(E1)] would be true if pre2(E1)=dc1, and [LatchOpen pre1(E1)] would be true if pre2(E1)=lo1. Adding these assumptions creates a plan that involves pulling the door after 8AM (since the Latch must be open) and before 9AM (to satisfy the

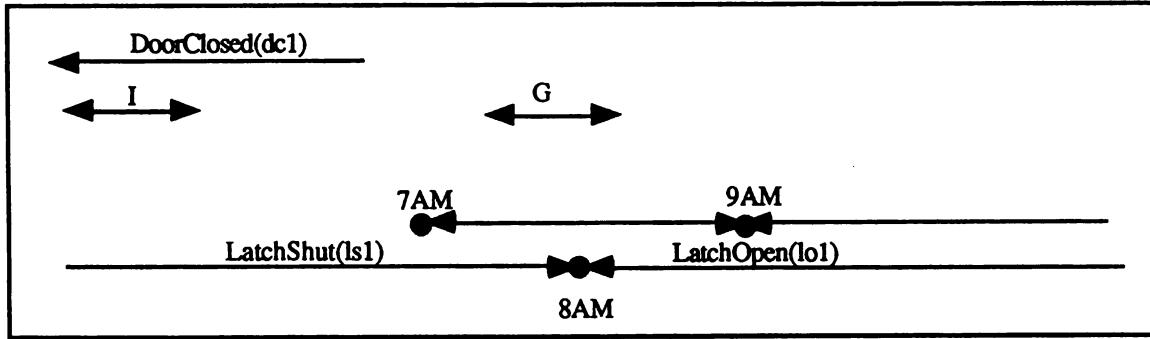


Figure 5: The door problem with an automatic latch system

goal conditions on G). Thus we have constructed a plan that takes advantage of the automatic latch, a known event in the future, by scheduling its own actions to take advantage of the latch being open. If, on the other hand, the goal had been to open the door before 8AM, i.e. G is constrained to be before 8AM, then this plan will not be suggested since the persistence assumption $\text{pre1}(E1)=lo1$ is not consistent with the database.

7 DISCUSSION AND EXTENSIONS

7.1 Persistence Assumptions

One of the critical techniques used to construct plans was the use of assumptions about the equality of times. In essence, this technique allows a time period to extend as far as possible given its constraints to other periods. Thus, if a proposition P is known to be true over an interval that contains a time t, and nothing is known to prevent P remaining true after t, then an assumption may be made that P is true after t if needed in a plan. Similarly, if nothing prevents P from being true before t, an assumption might be made that P was true some time before t. This capability to extend forwards or backwards in time might seem strange at first, but is quite useful in tasks that require plan recognition, or planning in worlds where information has to be acquired.

In Allen et al (1991), we show that this technique corresponds to the technique of Dean & McDermott [1987] if the persistence is into the future. The differences arise in two cases: first when there is uncertainty as to the starting time of the property, and second when a property is proven by extending backwards. Thus, the interval persistence rule is considerably more liberal than the rule used by Dean and McDermott. To handle these latter cases, Dean introduces another mechanism based on abduction. The different cases, however, seem to all reflect the same interactions, so a single uniform method for handling them seems preferable. In addition, the interval persistence rule is considerably simpler to describe and analyze. Situations requiring the more general rule appear frequently in everyday situations.

For example, if we are told that our airline tickets will be at the Bursars Office at 3 on Tuesday, then that suggests that they might be there earlier - it depends on the unknown information about when they were delivered. Similarly, being there at 3 suggests that the tickets will be there after 3 as well, and how long depends on unknown information about when they were picked up. In a single-agent domain, we have a high degree of confidence that the tickets remain at the office until we pick them up, since no other agent exists to pick them up. Note, of course, in a single agent domain, there wouldn't be an agent to deliver the tickets to the office in the first place, so that the tickets would need to be at the office in the initial situation. Thus, with a single agent, and a completely defined initial world, there is a strong bias to persistence only into the future. With multi-agent worlds, and partially defined situations, extension into the past becomes an equally important technique.

Of course, a simple persistence technique like this has problems. Using logical consistency is too weak a measure for the plausibility of a persistence assumptions. Rather, it would be better to evaluate the likelihood of a persistence using a causal theory, or a probabilistic method such as in Dean & Kanazawa[1988]. Note that since the assumptions are explicitly part of the final plan, such techniques could be introduced into this framework to produce a likelihood that a given plan will succeed if attempted.

7.2 A Hierarchical Planning Algorithm

The representation and algorithm above can easily be extended to include reasoning based on action decomposition as found in hierarchical planners such as NONLIN (Tate, 1977), SIPE (Wilkins, 1988) or FORBIN (Dean, Firby & Miller, 1989).

We could do this by introducing axioms that allow us to prove formulae of the form $[Try\ a\ e\ t]$ rather than assuming them. But it turns out that the algorithm is easier to specify if we introduce a new predicate *Decomp* on actions. The relation between *Decomp* and *Try* is

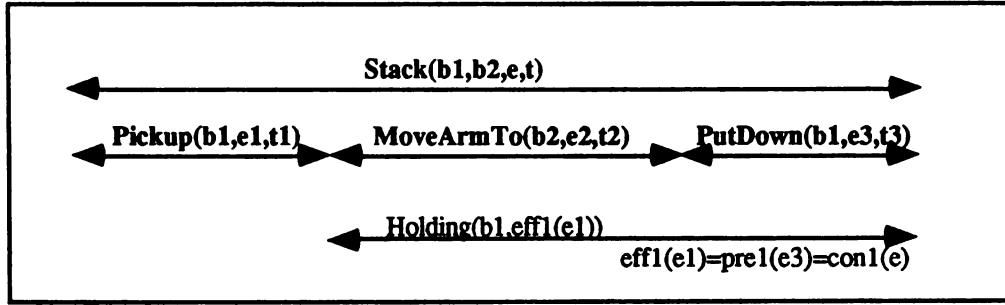


Figure 6: The decomposition of the Stack Action

that you try an action by performing one of its decompositions, ie.

$$\forall e, t, a . \text{Decomp}(a, e, t) \supset \text{Try}(a, e, t).$$

For example, a stacking action is accomplished by moving the arm to the block desired, opening the hand, lowering the hand over the block, grasping the block, raising the arm and moving it to the desired destination, and then lowering the arm and opening the hand again. Figure 6 shows the decomposition of the Stack action together with the necessary persistence assumptions required to make the decomposition effective. In particular, the effect of picking up b1, namely *Holding(b1,eff1(e1))*, must extend to satisfy the precondition on the PutDown action. In addition, this effect is identical to the constraint originally defined for the Stacking action.

We capture this information by adding a planning rule that specifies this as one way to decompose the action:

Stack Decomposition Axiom

[Decomp [stack ?x ?y] ?e ?t] <<< [Pickup ?x ?e1 ?t1]
[MoveArmTo ?y ?e2 ?t2] [PutDown ?x ?e3 ?t3]
such that [Meets ?t1 ?t2][Meets ?t2 ?t3][Starts ?t1
?t] [Finishes ?t3 ?t] [EQ eff1(?e1) pre1(?e3)]
[EQ eff1(?e1) con1(?e)].

The only other complication is that the initial algorithm used a second stage to verify that all persistence assumptions made in the plan were still consistent. We could leave this second stage until the entire plan is decomposed, but it is more in line with traditional hierarchical planners to verify these assumptions at each decomposition level before the next level is constructed. This can be accomplished simply in the new algorithm by adding a "dummy" goal on the bottom of the goal stack that invokes the consistency checking algorithm. When this goal rises to the top of the stack, one complete level of decomposition has been completed. The constraints are checked and the dummy goal is added again at the bottom of the goal stack to signal the end of the next level of decomposition. We will call this dummy goal [VerifyAssumptions]. A precise specification of the algorithm is as follows: This algorithm is a slight variation of the earlier algorithm. It differs in how action attempts are treated, and in the

time that assumptions are verified. As before, this is a non-deterministic version of the algorithm, and the goal stack GS is initialized to the goal statement.

Do until GS is empty:

- (0) Remove the top element of GS and call it G;
- (1) Choose
 - (1.1) If a formula unifying with G is found in the database, bind any variables in G as necessary.
 - (1.2) Otherwise, if G can be proven by a persistence assumption, then pass G to the prediction reasoner, and add G together with the equality assertion that justifies the assumption to the assumption list.
 - (1.3) Otherwise, if G is of the form *Try(A, e, t)*, for some action A, add G to the action list and pass G to the prediction reasoner. Also, if there are axioms with a consequence of form *Decomp(A, e, t)* in the database, add *Decomp(A, e, t)* to the end of GS.
 - (1.4) If G = [VerifyAssumptions], then invoke the assumption verifier. (Note, if verifying the assumptions fails, then G is not achieved and the algorithm backtracks). Unless GS is now empty, add a new goal [VerifyAssumptions] to the end of GS.
 - (1.5) Otherwise, find a planning rule R whose antecedent unifies with G, instantiates the rule as defined above and push the antecedents of R onto GS.

This algorithm expands a plan level-by-level through a decomposition hierarchy, validating the consistency of the plan at each level before the next level was addressed. Constraints imposed by the higher levels makes the accomplishment of the actions at the lower levels considerably easier.

8 CONCLUSIONS

We showed how traditional planning systems could be recast fairly directly as a specialized inference process on a temporal logic. By doing this, we have produced a framework that is much easier to understand and extend. By separating the temporal aspects of a plan from the procedural aspects of plan construction, for example, we found that even the simplest backwards chaining planning algorithm can generate non-linear plans. Similarly, a hierarchical planner can be generated by changing the set of assumptions about action attempts that the system is willing to make at any given time. As such, this work provides a uniform framework for examining many of the different planning frameworks developed to date.

While the actual system described duplicated the abilities of traditional planning algorithms, the situations that can be represented and reasoned about are more general than can be represented in a state-based model. In particular, we can reason about plans involving complex interactions between overlapping actions. It can reason about the effects of simultaneous actions that are not the effect of any one of the actions individually. The representation is limited, however, in representing partial interference between actions. This is because the current representation cannot explicitly capture the notion of possibility (as found in branching time models) and from the simple technique used for generating persistence assumptions.

By separating the domain reasoning from the plan construction algorithm, we have developed a general representation for reasoning about action that is independent of the particular application that is driving it. A plan recognition system could use the same action definitions in the same representation. Plan recognition can be viewed as just another specialized inference process on this same world representation.

More details on all these topics can be found in Allen et al (1991).

Acknowledgements

This work was supported in part by the Air Force Systems Command, RADC (now Rome Laboratory) and the Air Force Office of Scientific Research, under contract no. F30602-85-C-0008, and NSF grant N00014-90-J-1811.

References

- Allen, J.F. "Maintaining knowledge about temporal intervals," *CACM* 26, 11, 832-843, 1983.
- Allen, J.F. "Towards a general theory of action and time," *Artificial Intelligence* 23, 2, 123-154, 1984.
- Allen, J.F. and P.J. Hayes. "A common-sense theory of time," *Proc. IJCAI 85*, Los Angeles, CA 1985.

Allen, J.F. and J.A. Koomen, "Planning using a temporal world model", *Proc. IJCAI 83*, Karlsruhe, Germany, 1983.

Allen, J.F., H. Kautz, R. Pelavin, and J. Tenenberg *Reasoning About Plans* Morgan Kaufmann, 1991.

Bacchus, F., J. Tenenberg and H. Koomen, "A non-reified temporal logic", *Proc. of the First Int'l Conf. on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1989.

Davidson, D. "The logical form of action sentences," in N. Rescher (ed.). *The Logic of Decision and Action*. U. Pittsburgh Press, 1967.

DeKleer, J. "An assumption-based TMS," *Artificial Intelligence* 28, 127-162, 1986.

Dean, T., J. Firby and D. Miller. "Hierarchical planning involving deadlines, travel time and resources," *Computational Intelligence*, 1990.

Fikes, R.E., and N.J. Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence* 2 231-272, 1971.

Harel, D. "Dynamic logic," in *Handbook of Philosophical Logic, Vol. II*. Reidel, 1984.

Koomen, J.A. "Localizing temporal constraint propagation", *Proc. of the First Int'l Conf. on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1989.

McCarthy, J. and P. Hayes "Some philosophical problems from the standpoint of artificial intelligence" in *Machine Intelligence 4*. Edinburgh University Press, 1969.

McDermott, D. "A temporal logic for reasoning about processes and plans," *Cognitive Science* 6, 2, 101-155, 1982.

McDermott, D. "Reasoning about plans," in J.R. Hobbs and R.C. Moore (eds.). *Formal Theories of the Commonsense World*, Ablex, 1986.

Pelavin, R. "A formal approach to planning with concurrent actions and external events," TR 254, Computer Science Dept., U. Rochester, 1988.

Rosenchein, S.J. "Plan synthesis: A logical perspective," *Proc. IJCAI*, 331-337. Vancouver, British Columbia, 1981.

Sacerdoti, E.D. *A Structure for Plans and Behavior*. New York: American Elsevier, 1977.

Shoham, Y. "Temporal logics in AI: Semantical and ontological considerations," *Artificial Intelligence* 33, 1, 89-104, 1987.

Tate, A. "Generating project networks," *Proc. IJCAI*, 888-93, Cambridge, MA, 1977.

Vere, S. "Planning in time: Windows and durations for activities and goals," *IEEE Trans. Pattern Analysis Mach. Intell.* 5, 3, 246-67, 1983.

Waldinger, R. "Achieving several goals simultaneously", in Elcock, E & Michie, D (eds.), *Machine intelligence 8*, Ellis Horwood, pp 94-136.

Wilkins, D. *Practical Planning*, Morgan Kaufmann, 1988

Temporal Reasoning and Narrative Conventions

Jonathan Amsterdam
 MIT Artificial Intelligence Lab
 545 Technology Square
 Cambridge, MA 02139
 jba@ai.mit.edu

Abstract

Much of the confusion over the proper treatment of temporal reasoning problems such as the Yale Shooting Problem stems from a failure to appreciate the conventions that operate when these problems are communicated. Here I discuss one such convention and present a formal theory of temporal reasoning that incorporates it.

1 INTRODUCTION

Narrative conventions—implicit rules we all use in understanding stories—shape our intuitions about temporal reasoning problems. I present a novel theory of temporal reasoning that takes one such convention into account.

Temporal reasoning problems have the following structure: given a set of first-order formulas that describe both the rules of the domain and some of the facts of a particular situation, determine what happens before, during or after the situation. The classic example of a temporal reasoning problem is the Yale Shooting Problem (YSP) (Hanks and McDermott, 1987). In the YSP, we are given the information that a gun is loaded and then fired at someone who is alive, and we are told that firing a loaded gun at someone results in death. We are then asked if the target is dead after the gun is fired. The intuitive answer is yes, but early non-monotonic logics, e.g. (McCarthy, 1987; Reiter, 1987), also sanction the answer no, because they permit models in which the gun is unloaded between its loading and firing.

Since Hanks and McDermott's paper, many solutions have been proposed which correctly handle the YSP, although most fail in other ways on other temporal reasoning problems. More critically, few of these solutions have addressed the factors that shape our intuitions about temporal reasoning problems. Since these intuitions are central to any attempt to formalize

temporal reasoning—they are the data we are trying to explain—it would be useful to examine the factors that underly them. I claim that one such factor is the form in which these problems are presented.

2 NARRATIVE CONVENTIONS

Temporal reasoning problems can be written as a set of first-order sentences, but they are more commonly transmitted and discussed among researchers in this way:

Fred is alive. A gun is loaded. Some time passes. The gun is shot at Fred.

The first and most obvious thing to notice about this piece of text is that it's a story. In particular, it's a *narrative*: a story about a sequence of events. We may experience narratives through reading or in the course of conversation, and whenever we encounter them, we draw inferences about what is said and not said. Many of these inferences involve commonsense knowledge about the physical and social worlds, but some are bound up with the conventions we all follow implicitly whenever we are confronted with the narrative mode. By reflecting on these narrative conventions, even if superficially, we can gain a better understanding of temporal reasoning. Here I consider one such convention.

Re-read the above story, then ask yourself: "Where am I?" The answer seems to be that you are located where the action is. This feeling can be strengthened with a little rewriting:

Story 1. Bill loaded the gun while Fred stood there. There was a pause. Then he pointed the gun at Fred and pulled the trigger.

Here there is no question that the reader feels like a direct and attentive observer. In effect, the narrator serves as the reader's senses, describing just what the reader would notice if actually present. This property of narration, which I call the *You Are There* convention

(YATC for short), is ubiquitous and very powerful. A writer must go out of the way to avoid placing the reader in the action. Here I've made some changes to get the reader out of the scene:

Story 2. I came to in time to see Bill loading the gun. Fred just stood there. I blacked out again for a moment or two, and when I came around I saw Bill point the gun at Fred and pull the trigger.

Note that the change to first person makes it easier to take the narrator out of the scene. The YATC is strongest with a third-person omniscient narrator—one who is not a character in the story and who does not obtrude in any way on the events being recounted. We have a strong inclination to trust such a narrator and to assume that the narrator has ideal observational capabilities. Indeed, we may not even be aware that there *is* a narrator; we simply perceive the story as unfolding before us as we read. Temporal reasoning problems are typically presented with this sort of narration.

Authors' attempts to defeat the You Are There convention can be quite subtle. In Kleist's "The Marquise of O—" (Kleist, 1978), a woman finds herself pregnant with no recollection of having performed the prerequisite act. The man who admits to being the father has spent only a few moments with her, and You Are There:

He smashed the hilt of his sword into the face of one of the murderous brutes, who still had his arms round her slender waist, and the man reeled back with blood pouring from his mouth; he then addressed the lady politely in French, offered her his arm and led her into the other wing of the palace which the flames had not yet reached and where, having already been stricken speechless by her ordeal, she now collapsed in a dead faint. Then — the officer instructed the Marquise's frightened servants, who presently arrived, to send for a doctor; he assured them that she would soon recover, replaced his hat and returned to the fighting.

Here the author defeats the YATC with a stroke of the pen: the deed was done at the “—”.

If the YATC is not overridden, what can we say about the information implicitly contained in the narrative? The YATC implies that the narrator is acting as an ideal attentive observer for the reader. An ideal observer would notice anything of importance that happened. So the YATC implies that *if an action is not mentioned, it did not happen*. This bald statement needs to be qualified in several ways. Irrelevant actions, such as the next-door neighbor's having lunch, can of course be omitted, as can causal chains begin-

ning with a mentioned action, such as the hammer descending on the firing pin as a result of the trigger being pulled. But even in its crude form, the principle gives insight into the intuitions behind temporal reasoning problems.

Consider Story 1. The YATC explains why the model in which the gun is unloaded seems so absurd: we are simply not told that the gun was unloaded. To see how strong the convention is, consider the “backwards YSP,” where Fred lives:

Story 3. Bill loaded the gun while Fred stood there. There was a pause. Then he pointed the gun at Fred and pulled the trigger. Nothing happened. Fred smiled.

What is the explanation for Fred's survival? Perhaps Bill loaded a blank, or his aim was off, or maybe Fred is a ghost. There are lots of possibilities, but one of them certainly is *not* that the gun was unloaded during the pause. It *couldn't* have been unloaded, or you would have been told.

Now let's consider the YSP with the YATC defeated (Story 2 above). Here we again predict Fred's death, but we don't balk quite so much at the idea that the gun was unloaded. When we're told that Fred survived the shot, we have no problem with Bill's unloading the gun.

2.1 NARRATIVE VS. REAL LIFE

Why go through the gyration of describing attentive observation in terms of a narrative convention? Why not just talk about agents actually in situations and dispense with a discussion of narrative?

There are two reasons. First, in the actual course of research and discussion, temporal reasoning problems are communicated as narratives, and this shapes our intuitions. Second, and more subtly, if we are to accept the working hypothesis that low-level sensory modules deliver information to our reasoning apparatus as logical sentences (as we must if we are to assume that agents confronted with real-world situations reason about temporal reasoning problems in the form we have been writing them), then the form of that information is much closer to narrative than to “raw” experience. For instance, much that is irrelevant (as determined by low-level mechanisms) will be omitted. In effect, an agent's sensory modules act as narrators, culling important information for transmission to higher-level reasoning centers. Hence narrative conventions can be said to play a role even in direct experience.

2.2 MORAL

We have observed that the YATC can affect our intuitions about temporal reasoning problems like the

YSP. By default, the convention is in force, and we conclude that actions not described do not occur. Only if the convention is explicitly overridden are we willing to hypothesise other actions. Failure to appreciate the convention has manifested itself in the field in two ways. First, accepted answers to temporal reasoning problems fail to consider one possibility, either the case where the YATC is in force or the case where it isn't. For example, an accepted intuition about the backwards YSP is that the gun is unloaded by an UNLOAD action. But the correct conclusion, given that the YATC is in force and no other causal rules are provided, is that the theory is inconsistent, or that the gun unloads "magically," with no action. Second, no one has ever attempted to express the distinction between a narrative where the YATC is in force and one where it isn't in any existing formalisation.

3 THE THEORY

Recognising the importance of narrative conventions does not require jettisoning current wisdom on how to do temporal reasoning. On the contrary: my theory makes heavy use of concepts like causality and explanation that play a key role in existing theories. One way in which the theory differs significantly from others is in providing a means to express when the YATC is in force, and in using this information.

I base my theory on the intuition that a good solution to a temporal reasoning problem is a *good explanation* of the phenomena described in the problem. A good explanation should avoid hypothesising change unless necessary, and should provide reasons, in the form of action occurrences, for any changes that do occur.

To formalise the theory, I view an explanation as a proof in the object language. The proof may proceed entirely from given information—facts about the situation and causal rules—or (in the interesting cases) it may hypothesise additional information. My theory sanctions certain hypotheses: they are *supported*. Hypotheses that are not supported must be assumed. I judge an explanation by the assumptions it makes. Like other formal theories of temporal reasoning, I am performing a minimisation: I am minimising assumptions.

Let me first describe the notion of support in English. Section 3.1 contains the formal analogue of this description. A statement is supported in a situation if it is true in that situation, and one of the following holds:

1. The statement is given in the problem description.
2. The statement describes the non-occurrence of an action. A statement that nothing happens requires no additional justification.
3. The statement describes the occurrence of an ac-

tion, and this action is the only one which can cause a state change. For instance, the unloading of a gun is supported if the gun's state has changed from loaded to unloaded, and no other action can explain this. Since there are no uncaused events, an action that causes an event is justified by its ability to explain that event.

4. The statement describes a state which is the same as the state at the immediately preceding (or following) time-point. This is the notion of *persistence*: things do not change, all else being equal. The actual definition is somewhat more complex, in order to capture the "all else being equal." First, I require that the statement describing the state at the adjacent time-point also be supported. Without this addition, adjacent states could support each other; instead, support must be inherited forward or backward in time from an initial seed. Second, we must be able to justify the argument that the negation of the state change whose support is at issue cannot occur. If the negation could occur, then there is reason to question the persistence. The formal elaboration of this point is somewhat complex, and is presented below.
5. The statement is provable from other supported statements. It is important that any notion of justification embody ordinary first-order provability, since provability is at the heart of justification. For instance, we would not know what to make of a theory in which p and $p \rightarrow q$ could be justified and q unjustified.

3.1 DEVELOPMENT

Now I proceed to the formal development. I adopt a model of time isomorphic to the integers and use a propositional temporal logic, writing p_t to say that proposition p holds at time t . I use upper-case for propositions expressing the occurrence of actions, as in SHOOT_t .

A *ground time-statement* (GTS for short) is a statement of the form φ_t , where φ is a possibly negated proposition or action and t is a time-point, i.e. integer. A ground time-statement asserts the occurrence or non-occurrence of an action, or the truth or falsity of a proposition, at a particular instant of time.

To allow the YATC to be overridden, I introduce an operator U on formulas, and the inference rule $U\varphi \rightarrow U\neg\varphi$. U is applied to action occurrences and expresses the fact that whether or not φ occurred is unknown, i.e. that the YATC is not in force. For instance, the sentence "I blacked out for moment" could be rendered as the set of sentences $U\alpha_t$, where t is the time-point of the blackout and α ranges over the possible actions. The U operator does not affect the standard (first-order, monotonic) semantics of the lan-

guage. Statements preceded by \mathbf{U} are ignored in determining the models of a set of formulas. \mathbf{U} is used only in the non-monotonic portion of the theory.

I express a temporal reasoning problem as a set Σ of first-order sentences, divided into a *domain theory* containing general rules and a *chronicle description* providing specific facts, including \mathbf{U} formulas. The chronicle description corresponds to the narrative. Figure 1 shows the YSP in this notation, with no \mathbf{U} formulas. (Universal quantification is implicit.)

```

Chronicle description
alive1
loaded1
SHOOT2

Domain theory
UNLOADt → ¬loadedt+1
SHOOTt ∧ loadedt → ¬alivet+1
```

Figure 1: The Yale Shooting Problem

To deal with causality, I assume a meta-level function $\text{Effects}(\mathbf{A}_t, \beta)$ associated with each domain theory T , where \mathbf{A} is an action, t is a time point, and β is a (possibly infinite) set of GTSs (the preconditions). The value of the function is a set of GTSs describing all the effects that must obtain when \mathbf{A}_t and β obtain, given T . Effects could actually be “implemented” in a variety of ways. For instance, we could introduce a special kind of formula called a “causal rule” into the object language and compute Effects from the set of causal rules, or we could introduce an Effects predicate into the object language. (Either of these choices would also require some sort of closed-world assumption.) By making Effects a function at the meta level, I am trying to be noncommittal about how the effects of actions are actually specified.

Given Effects , I can define causation. In the following definitions, I assume a fixed set Σ of sentences.

An action occurrence \mathbf{A}_t causes a proposition φ_t in a model M of Σ iff there is a set of GTSs β (the preconditions) such that $M \models \mathbf{A}_t$, $M \models \beta$ and $\varphi_t \in \text{Effects}(\mathbf{A}_t, \beta)$. A proposition is caused if there is an action occurrence that causes it.

\mathbf{A}_t potentially causes φ_t (and is a potential cause of φ_t) in a model M of Σ iff \mathbf{A}_t does not cause φ_t in M but there are other models of Σ in which \mathbf{A}_t causes φ_t . A blocking statement for a potential cause in a model M is a GTS entailed by M that prevents causation (e.g. $\neg\mathbf{A}_t$ or a negated precondition). In other words, an action potentially causes an event if it would have caused that event had certain circumstances (the blocking statements) been different.

A proposition is uncaused if it is not caused; supportedly uncaused if it is uncaused and for each potential

cause, at least one blocking statement is supported.

I now turn to the heart of the theory, the definition of support. Given a set of sentences Σ , a model M of Σ and an assumption set AS of GTSs, where $M \models AS$, I define the set of supported statements, or *supported set*, of M and AS as the least set S that contains all GTSs φ such that $M \models \varphi$ and for which one of the following holds:

1. (*Given.*) $\varphi \in \Sigma$ or $\varphi \in AS$.
2. (*Non-occurrence.*) φ is of the form $\neg\mathbf{A}_t$.
3. (*Explaining state-change.*) φ is of the form \mathbf{A}_t , $M \models p_{t-1} \wedge \neg p_t$, and \mathbf{A}_t is the only action occurrence which causes $\neg p_t$ in M .
4. (*Forward persistence.*) φ is of the form p_t , p_{t-1} is supported, and $\neg p_t$ is supportedly uncaused.
5. (*Backward persistence.*) φ is of the form p_t , p_{t+1} is supported, and $\neg p_{t+1}$ is supportedly uncaused.
6. (*Provability.*) $S \vdash \varphi$.

Throughout the above definition, \mathbf{A} stands for an action proposition and p a possibly negated state proposition. A GTS is *supported* (given Σ , M and AS) iff it is in S . The definition of S is monotone; it is never the case that adding a statement to S requires the removal of other statements from S . This implies that there is a unique S .

At this point, it would be worthwhile to compare this six-point definition with the five-point English gloss of the previous section. (The English clause 4 has been split into two clauses here for readability.) The only substantive difference is the presence of AS in clause 1 above; this was done to simplify the definition.

An assumption set AS is *admissible* for M iff every GTS which M entails is supported. An admissible AS , then, contains the assumptions needed to justify every event in M .

A model M is *admissible* iff: (1) every state change is caused and (2) for every action occurrence \mathbf{A}_t such that $M \models \mathbf{A}_t$, either $\Sigma \vdash \mathbf{A}_t$ or $\Sigma \vdash \mathbf{U}\mathbf{A}_t$. This is how I incorporate the YATC: I do not allow hypothesised actions unless it is known that they could have occurred (i.e. the YATC is not in force for them). If no models are admissible, then Σ is inconsistent (according to the metalogic).

Following the usual practice, I first define a preference criterion on admissible models, then endorse those models that are minimal in the criterion. I prefer one model to another if the first has a strictly better explanation than the second. The formal translation of this is, roughly, that the first model has an admissible assumption set that is a subset of all the admissible assumption sets of the second. In fact, I wish to treat a statement and its negation as the same. So given a set of statements α , define $\bar{\alpha}$ as $\{\varphi \mid \varphi \in \alpha \vee \neg\varphi \in \alpha\}$,

write $\mathcal{A}(M)$ for $\{\alpha \mid \alpha \text{ is an admissible assumption set of } M\}$, and prefer M_1 to M_2 iff there is an $\alpha \in \mathcal{A}(M_1)$ which is a subset of all the members of $\mathcal{A}(M_2)$.

3.2 EXAMPLES

3.2.1 The YSP

First, let's consider the YSP, as shown in Figure 1. The Effects function is obvious from the domain theory:

$$\begin{aligned}\text{Effects}(\text{UNLOAD}_t, \{\}) &= \{\neg\text{loaded}_{t+1}\} \\ \text{Effects}(\text{SHOOT}_t, \{\text{loaded}_t\}) &= \{\neg\text{alive}_{t+1}\}\end{aligned}$$

The GTSs for this problem include UNLOAD_t , SHOOT_t , alive_t and loaded_t and their negations, for all t . As presented in Figure 1, the YATC is in force.

First consider the “correct” models, in which $\neg\text{alive}_2$ and the only action is SHOOT_2 . All the GTSs for these models are supported with the empty assumption set:

- All action non-occurrences, including $\neg\text{UNLOAD}_1$, are supported by clause 2.
- loaded_t is supported for all t by persistence. In particular, loaded_2 is supported by clause 4, because (a) loaded_1 is supported (by being in Σ), and (b) $\neg\text{loaded}_2$ is supportedly uncaused, because any potential cause is blocked by $\neg\text{UNLOAD}_1$, which is supported.
- $\neg\text{alive}_2$ is supported in virtue of being provable from the SHOOT action and loaded_2 .
- All other GTSs, namely alive_t and $\neg\text{alive}_t$, $t > 3$, are supported by persistence.

With the YATC in force, models which contain any other action are not admissible by part 2 of the definition of admissible. So the correct models are the only admissible ones.

Now consider the YSP with the YATC defeated at $t = 1$, which we obtain by adding the statements $\text{U}(\text{UNLOAD}_1)$ and $\text{U}(\text{SHOOT}_1)$ to Figure 1. It is still the case that the correct models are preferred, because any other model requires at least one assumption. In particular, consider a model in which UNLOAD_1 occurs. If this action is not an assumption, it must be supported by the state change from loaded to unloaded, by clause 3 of the definition of support. For $\neg\text{loaded}_2$ to be supported, it must either be assumed or it must follow from alive_2 . (It could also be supported via backward persistence from $\neg\text{loaded}_3$, but this would lead us down the same path of reasoning for $\neg\text{loaded}_3$.) But for alive_2 to be supported and not assumed, it must follow by persistence from alive_1 . This can only happen if $\neg\text{loaded}_2$ is supported, because it is the sole blocking statement of a potential cause, namely SHOOT_2 . So at least one assumption is necessary.

Chronicle description

```
alive1
loaded1
SHOOT3
alive4
U(SHOOT1)
U(SHOOT2)
U(UNLOAD1)
U(UNLOAD2)
```

Domain theory

```
UNLOADt → ¬loadedt+1
SHOOTt ∧ loadedt → ¬alivet+1
```

Figure 2: The Backwards YSP with YATC suspended

3.2.2 The Backwards YSP

For a second example, we consider a version of the YSP with two changes: the time of the shooting is changed to $t = 3$ and alive_4 is asserted. (See Figure 2.) This is the backwards YSP. In this version, the YATC is suspended for times 1 and 2 (modeling, e.g., the statement “I blacked out for a moment”). The answer we would like is that a single UNLOAD occurred, either at time 1 or 2.

From the given information we can prove $\neg\text{loaded}_3$. The only admissible models have either UNLOAD_1 or UNLOAD_2 , or both. (These models are admissible because $\text{U}(\text{UNLOAD}_1)$ and $\text{U}(\text{UNLOAD}_2)$ are provable from the given information.) Models with $\neg\text{UNLOAD}_1$ support loaded_2 by persistence and UNLOAD_2 by explanation of state change, with no assumptions, and similarly for models with $\neg\text{UNLOAD}_2$, so these two classes of models are preferred. Models with both UNLOADs are not preferred because only one of the UNLOADs can explain the state change; the other must be assumed. This is the desired answer.

Finally, we examine the backwards YSP with the YATC in force. This problem is given by Figure 2 with the U formulas omitted. Now no models are admissible, because any admissible model must have either UNLOAD_1 or UNLOAD_2 in order for the loaded-to-unloaded state change to be caused, but neither of these action occurrences is provable from the problem. (Recall that for a model to be admissible, it must be the case that all state changes are caused and also that Σ must be able to prove, for every action occurrence A_t , either A_t or $\text{U}(A_t)$.) We can interpret this as saying that the story is inconsistent (according to the metalogic), or that a “magical,” uncaused state change must occur.

3.3 LIMITATIONS OF THE THEORY

The theory I have outlined is only a first step. Several extensions could be made to cover a wider class of situa-

ations. For instance, there is no provision for handling cases where multiple simultaneous actions are required to achieve a result. Also, I have given no criteria that prefer one action to another when more than one action can explain a state change. Natural suggestions are to prefer an action which causes the fewest additional state changes, and to prefer an action which can also explain other uncaused state changes.

The theory's greatest limitation is its overly rigid interpretation of the YATC: any action not mentioned in the narrative (or provable from it) cannot occur. Recall that in section 2, I said that some actions, like those in the middle of causal chains, need not be mentioned in the narrative even when the YATC is in force. The theory fails on certain temporal reasoning problems with causal chains, concluding that the intuitively correct models are inadmissible because they contain actions that are not provable from the problem statement. The theory also fails for the same reason on cases where an action is obvious even though it is not mentioned. For instance, if a character is sitting by the fireplace in one sentence and standing by the door in the next, then clearly some movement occurred. Depending on the wording of the narrative, a reader might not demand that the narrator describe the obvious intervening action.

Modifying the formal theory to handle these issues would at the least require distinguishing between different kinds of support. For example, statements supported in virtue of their being parts of a causal chain would need to be distinguished from statements whose support derived from persistence. But before trying to adjust the formal theory, we should pursue a much more thorough account of what readers are willing to let the narrator get away with in the presence of the YATC.

4 RELATED WORK

My theory is closest to those of (Geffner, 1989; Geffner, 1990) and (Morgenstern and Stein, 1988). Like Geffner, I consider explanation crucial, but I treat the roles of object language and metalanguage differently. The theory in (Geffner, 1989) distinguishes between object language statements with different syntax. In (Geffner, 1990), ab terms representing abnormalities, and a special "explains" operator C, are introduced into the object language; statements about causality and persistence must be formulated carefully in terms of these additions. Since I view formalization as a tool for clarifying intuitions, I have tried to keep the object language simple as is feasible, at the expense of complicating the metalanguage.

My theory was largely inspired by Morgenstern and Stein's Motivated Action Theory (MAT). MAT recognises the importance of causality, but not of explanation. Morgenstern and Stein's concept of motivation

is like support, but is directed forward in time: an action can be motivated if known past events force it to occur, but not if known future events imply its previous occurrence. This asymmetry causes problems for their theory which my time-symmetric theory avoids.

For instance, consider the backwards YSP of Figure 2. By MAT, and by my theory, we must have at least one of UNLOAD₁ and UNLOAD₂. Both theories sanction only those models where exactly one of the two UNLOADs occurs. But while in my theory, each UNLOAD is supported, in MAT, both UNLOADs are unmotivated. MAT's result is somewhat surprising when a "compressed" version of the backwards YSP is considered, in which there is time for only one UNLOAD. In this case, the single UNLOAD action is motivated (in fact, provable). Although MAT still gets the right answer for this problem, it would fail if the UNLOADs had ramifications. This flaw in MAT stems from the forward-directedness of motivation.

5 CONCLUSION

Temporal reasoning problems are subject to the conventions that govern the understanding of narratives, in particular the convention that the narrator serves as an attentive observer for the reader. I have argued that failure to appreciate this convention has been the source of some confusion in understanding the intuitions brought out by temporal reasoning problems, and I have presented a formal theory that takes the convention into account.

While my theory shares with most theories of temporal reasoning a desire to minimise change, it departs from them in explicitly dealing with the structure of the explanation. Whatever flaws my particular theory may have, I believe it is at least on the right track, because it evaluates models not by what happens in them, but by how well we can explain what happens.

Acknowledgements

Thanks to Lynn Stein for considerable help with all aspects of this work, to David McAllester and Ronald Rivest for clarifying some technical points, and to Abigail Schweber for editorial assistance. Randall Davis and Gerald Sussman provided useful comments and criticism on an earlier draft.

References

- (Geffner, 1989) Hector Geffner. Default reasoning, minimality and coherence. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 1989.
- (Geffner, 1990) Hector Geffner. Causal theories for nonmonotonic reasoning. In *Proceedings of the AAAI*, 1990.

(Hanks and McDermott, 1987) Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

(Kleist, 1978) Heinrich von Kleist. *The Marquise of O & Other Stories*. Penguin Books, Harmondsworth, UK, 1978.

(McCarthy, 1987) John McCarthy. Circumscription—a form of nonmonotonic reasoning. In Matthew L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, Los Altos, CA, 1987.

(Morgenstern and Stein, 1988)

Leora Morgenstern and Lynn Andrea Stein. Why things go wrong: A formal theory of causal reasoning. In *Proceedings of the AAAI*, 1988.

(Reiter, 1987) Ray Reiter. A logic for default reasoning. In Matthew L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, Los Altos, CA, 1987.

Reflections about Reflection

Giuseppe Attardi
 Dipartimento di Informatica
 Università di Pisa
 Corso Italia 40, I-56125 Pisa, Italy

Maria Simi
 Dipartimento di Matematica e Informatica
 Università di Udine
 Via Zanon 6, I-33100 Udine, Italy

Abstract

In the line of a syntactic treatment of modalities, many proposals have been presented consisting in an extension of a logic calculus with a meta-language for expressing facts about terms and sentences, and with an axiomatization of provability. These proposals differ in the use of the inference rules used to link the object level and meta-level theories; the formulation of these reflection rules is crucial since it affects the consistency of the extended calculus. We argue that reflection rules resulting in a conservative extension are too weak. On the other hand, well known results show that non conservative extensions often run into paradoxes. We propose a non conservative extension where useful theorems can be proved while consistency is retained.

1 INTRODUCTION

The advantages of a syntactic approach to the representation of truth, knowledge and belief have been largely discussed in the literature [(McCarthy 1979), (Moore, 1977), (Konolige, 1982), (Perlis, 1985), (Perlis, 1988), (Davies, 1990)]. Along this line of research several proposals are based on extending a logic calculus with a meta-language, for expressing facts about terms and sentences, and with an axiomatization of provability. Provability in an agent's theory is seen as one of the primitive relations for the formalization of belief and knowledge.

These proposals differ in the degree of connection between object-theory and meta-theory, ranging from a semantic connection of completely separate theories as in

(Konolige, 1982), to the reflection principles of FOL (Weyhrauch, 1980), which provide a bridge between the two, still distinct, theories, to the amalgamated solutions, where there is just one theory encompassing object and meta-level.

In the "separated" approach, in order to express nested belief, a hierarchy of languages has to be constructed, where each language is a meta-language for the theory below in the hierarchy. Self reference is not allowed and the construction of paradoxical sentences is blocked. As a consequence also non paradoxical self referential or mutually referential sentences cannot be represented. This is seen as the major drawback of this approach (see for example (Perlis, 1985), (Perlis, 1988), (Davies, 1990)): self referential statements about truth, belief or knowledge are in fact naturally found in common sense reasoning.

Among the amalgamated solutions we can further distinguish among conservative extensions (like the one proposed by Bowen and Kowalski (Bowen and Kowalski, 1982) and non conservative ones such as the Omega extension in (Attardi and Simi, 1984)). In the case of amalgamated solutions, a proper formulation of the reflection rules is crucial since it affects the consistency of the extended calculus.

Reflection rules resulting in a conservative extension are very weak; in principle they do not add anything to the reasoning that can be performed in the object level or meta-level theories. On the other hand, well known negative results (Montague, 1963) show that non conservative extensions often run into paradoxes.

We will propose a formulation of the inference rules which lays in the middle ground between the Bowen-Kowalski proposal and the Omega proposal, resulting in a non conservative but consistent extension.

In section 2 we set the ground for the following discussion making precise our assumptions and defining the conventions to be used in the rest of the paper. In sections 3 and 4 we present an analysis of different formulations of the reflection rules, especially in the context of the amalgamated approach, and discuss the implications on the resulting theories. In section 5, 6 and 7 we propose a new version of the reflection rules, demonstrate with an example its expressiveness, and prove the consistency of the resulting logic. Finally, in section 8 and 9, we discuss related work and comment on the results.

2 META-APPARATUS AND BASIC ASSUMPTIONS

In the following we will assume that the meta-language is a first-order language whose domain of discourse are the syntactic expressions of the object language. We rely on the existence of a naming device for terms and sentences of the object language. Any of the mechanisms proposed in the literature will do, such as having terms of the meta-language for terms and sentences of the object language. We will use the convention of quoting expressions when their names should be used.

Moreover, the meta-theory shall include an axiomatization of provability as in (Bowen and Kowalski, 1982). This can be done, for example, by considering a set of inference rules in the style of natural deduction and writing a meta axiom for each of the inference rules. For instance, for the rule of implication introduction, formulated as follows:

$$\frac{\Gamma \cup A \vdash B}{\Gamma \vdash (A \Rightarrow B)}$$

we will have a corresponding meta axiom:

$$PR(\Gamma \cup A, B) \Rightarrow PR(\Gamma, A \Rightarrow B)$$

Here and in the rest of the paper we use the conventions that $\Gamma, \Gamma_1, \Gamma_2$ are metavariables for set of statements and A, B are metavariables for statements. In correspondence, ' Γ ', ' Γ_1 ', ' Γ_2 ' are sets of quoted statements and ' A ', ' B ' are quoted statements. Finally, \vdash_T means derivable according to the axioms and rules of the calculus T .

Given the different meanings in which the term "conservative extension" has been used in the literature, we feel the need to make precise its use in the context of this paper. Let us call OT the object level theory and MT the corresponding meta-level theory including an axiomatization of provability (the language of the two theories may or may not be the same); with RR we will

denote the reflection rules used to connect the two theories above. We will speak about "conservative" or "non conservative" with respect to the extension of the theory OT+MT as a result of adding the reflection rules, i.e. OT+MT+RR is a conservative extension of OT+MT when any theorem derivable from OT+MT+RR is also derivable from OT+MT, or in other words RR are derived inference rules and as such do not add new theorems.

This use complies with the classical definition given for example in (Shoenfield, 1967) according to which an extension is conservative if any formula of the original theory which is a theorem of the extension is also a theorem of the original theory. In our case the language of the original theory and of the extension are the same.

Different seems to be the use in (Bowen and Kowalski, 1982) who talk about the amalgamation being a conservative extension in the sense that "*no new theorems are provable in the amalgamation that were not already provable in either L or M*". We believe that this definition is too strong. In fact their own amalgamation is not conservative according to this definition: given a formula A provable in L but not in M and a formula B provable in M but not in L, the formula $A \wedge B$ is provable in the amalgamation but neither in L nor in M. The Bowen and Kowalski amalgamation is however a conservative extension according to our definition.

Still another perspective is taken, in the discussion about reflection presented in (Giunchiglia and Smaill, 1989). While we talk about extensions of the theory OT+MT through the addition of reflection rules, they talk about extensions of OT with meta axioms or inference rules.

3 SEPARATION OF THEORIES

In Konolige's so called "syntactic approach" (Konolige, 1982), propositional attitudes are expressed in a meta-language (ML) as relations between agents and sentences of the object language (OL). The meta theory includes an axiomatization of truth and of the provability relation in the object level theory.

In order to deal with nested beliefs, a hierarchy of such languages and theories has to be constructed and a complex technical machinery is required for linking the levels appropriately. Each theory in this hierarchy includes the language of the theory below, the language for talking about it, and an axiomatization of truth and provability. The reasoning of an agent corresponds to the reasoning in one of these meta theories, at the appropriate level according to the degree of nesting of his beliefs; for

example three levels are needed for the reasoning of the third wise man in the three wise men puzzle.

In this proposal the connection between meta-level and object level theory is a semantic one: determining the truth of a formula involving PR in the ML can be performed by "semantic attachment", i.e. performing a proof in the OL. This semantic connection can be exploited to simplify proofs and only when complete information on the object level theory is available; in principle however, all the reasoning can be performed at the meta-level using the "simulated" version of provability.

The major drawback of this approach is the lack of expressiveness. Since there is a different truth and provability predicate for each level, sentences that do not belong specifically to one level, such as "the agent has a false belief", are not naturally expressed.

In Weyrauch's FOL (Weyrauch, 1980) the beliefs of each agent is represented as a pair of OL/ML theories connected by "reflection principles"; the reflection rules are expressed as follows:

(in T)	A	<i>without dependencies</i>	
(in META)	<u>Theorem('A')</u>		(reflect up)
(in META)	<u>Theorem('A')</u>		(reflect down)
(in T)	A		

where META is the meta-theory of T, including as theorems all the facts that are assumptions in T, and an axiomatization of provability. The rules can be phrased as follows, in our notation:

$$\frac{\vdash_T A}{\vdash_{MT} PR('A')} \quad (\text{reflect up})$$

$$\frac{\vdash_{MT} PR('A')}{\vdash_T A} \quad (\text{reflect down})$$

MT is the meta theory of T. These rules require that A has been derived with no dependencies (A is a theorem of T) in order to be able to reflect it up.

In FOL, theories are kept separate, no amalgamation is attempted, and therefore it is an obvious fact that no theorems across theories are generated.

The approach to nested beliefs is not elegant. For example, in the solution based on FOL to the three wise

men puzzle proposed in (Aiello, Nardi and Schaefer, 1989), each agent is represented as a structure which includes three object level theories, for representing the knowledge of the other two agents and himself, plus a meta-level theory where all the reasoning is done. This is necessary in order to maintain separate between the object level facts which represent the beliefs of each agent, including the agent itself.

An evolution of the ideas of FOL is the work on multi-language systems (Giunchiglia and Serafini, 1990). A multi-language system includes a set of distinct languages each associated with its own theory. These theories are not independent of one another but connected by "bridge" rules, whose premises and conclusions belong to different languages. For example reflection rules, which are the most typical example of bridge rules, could be expressed as follows:

$$\frac{<A; i>}{<PR('A'); j>} \quad (\text{reflect up})$$

$$\frac{<PR('A'); j>}{<A; i>} \quad (\text{reflect down})$$

where $<A; i>$ represents the formula A in the language i .

This approach has the flavour of general framework more than of a specific theory: no commitment is made to any particular formulation of the reflection rules, nor to the fact that the languages are necessarily distinct. This makes it difficult to make a comparison with other proposals. All we can say is that we seem to remain in the realm of layered approaches with the shortcomings in expressiveness we have discussed above.

4 AMALGAMATION OF LANGUAGE AND META-LANGUAGE

In the amalgamated approach we have just one theory including both the object level and meta-level theories, connected by inference rules. The interesting case is when the meta-language is the same as the object language ($M = L$); we will write both \vdash_L and \vdash_M as \vdash meaning derivability in the amalgamated theory. The amalgam includes an axiomatization of the provability predicate PR .

With these assumptions, the inference rules for Bowen-Kowalski's amalgamation become:

$$\frac{\Gamma \vdash A}{\vdash PR(\Gamma, 'A'))} \quad (1 - \text{reflect up})$$

Proof:

$\vdash_{PC} (R \Rightarrow \neg R) \Rightarrow \neg R$	(tautology of PC)
$\vdash_{PC} (PR(\neg R) \Rightarrow \neg R) \Rightarrow \neg R$	(Lemma 4)
$\vdash PR((PR(\neg R) \Rightarrow \neg R) \Rightarrow \neg R) \Rightarrow \neg R'$	(iii)
$\vdash PR(PR(\neg R) \Rightarrow \neg R) \Rightarrow \neg R'$	(ii)
$\vdash PR(\neg R)$	(iv)
$\vdash \neg R$	(i)
$\vdash R$	(Lemma 4)

■

Corollary. The amalgamated logic with the inference rules RR2 is inconsistent.

The theorem $PR(A) \Rightarrow A$ together with the inference rule

$$\frac{\vdash A}{\vdash PR(A)}$$

also lead to inconsistency. In fact it is easy to show that conditions (i)-(v) of Theorem 2, can be derived (Montague, 1963).

The restrictions introduced by RR3 reflection rules are enough to prevent application of Theorem 2. In fact Montague and Kaplan have shown that if any of the hypothesis (i)-(v) is dropped, Theorem 2 does not hold any longer. In our case, condition (i) follows from Lemma 1, condition (iii) is the weak version of reflect up, and condition (iv) is derivable with any version of the reflection rules. Condition (ii) however can be proved with rule (2-reflect up) but not with the weaker version (3-reflect up). Condition (ii) is not true for some sentences and therefore cannot be assumed as an axiom if we want to preserve consistency. This leads to the following:

Theorem 3. The amalgamated logic RT is consistent.

Proof: To prove consistency, we show that the logic RT can be translated into the logic TLN , which is a subset of the logic $S4LN$ (Davies, 1990) proved consistent from results in (Turner, 1989). As a consequence RT is consistent.

The logic TLN is a modal logic of truth, whose language includes the truth predicate T and the falsity predicate F . The axiomatization of TLN consists of standard axioms for first-order logic with equality and the following axioms for truth:

IP	$T('A \Rightarrow B') \Rightarrow (T('A') \Rightarrow T('B'))$
T	$T('A') \Rightarrow A$
BAR	$\forall x T('A') \Rightarrow T(' \forall x A')$
LNEC	$\frac{\vdash_{PC} A}{\vdash T('A')}$

The translation between RT and TLN is defined recursively as follows:

$$\begin{aligned}\tau(A) &= A, \text{ if } A \text{ does not contain } PR \\ \tau(PR(A', B')) &= T(' \tau(A) \Rightarrow \tau(B)) \\ \tau(A \wedge B) &= \tau(A) \wedge \tau(B) \\ \tau(\neg A) &= \neg \tau(A) \\ \tau(\forall x A) &= \forall x \tau(A)\end{aligned}$$

We must show that all axioms and inference rules of RT hold in TLN . Considering 3-reflect up, we must show that:

$$\frac{\Gamma \vdash_{PC} A}{\vdash T(' \wedge \Gamma \Rightarrow A')}$$

From the premise we obtain $\vdash_{PC} (\wedge \Gamma \Rightarrow A)$ by implication introduction, and hence we obtain the conclusion by means of LNEC.

For (3-reflect down), which is equivalent to Lemma 1, we need to show:

$$T(' \wedge \Gamma \Rightarrow B') \Rightarrow (\wedge \Gamma \Rightarrow B)$$

which is just an instance of axiom T.

All the axioms of RT which correspond to the axiomatization of provability, hold in TLN . Consider for instance the meta-level counterpart of Modus Ponens. We need to show that:

$$T(' \wedge \Gamma \Rightarrow A') \wedge T(' \wedge \Gamma \Rightarrow (A \Rightarrow B)) \Rightarrow T(' \wedge \Gamma \Rightarrow B')$$

Since

$$\vdash_{PC} (\wedge \Gamma \Rightarrow (A \Rightarrow B)) \Rightarrow ((\wedge \Gamma \Rightarrow A) \Rightarrow (\wedge \Gamma \Rightarrow B))$$

from LNEC we obtain

$$T('(\wedge \Gamma \Rightarrow (A \Rightarrow B)) \Rightarrow ((\wedge \Gamma \Rightarrow A) \Rightarrow (\wedge \Gamma \Rightarrow B)))$$

and then, using K

$$T('(\wedge \Gamma \Rightarrow A) \Rightarrow (\wedge \Gamma \Rightarrow B))$$

Finally, using K again,

$$T(' \wedge \Gamma \Rightarrow B')$$

Analogously, for any other inference rule and axiom it can be shown that their translation is provable in RT . ■

$$\frac{\vdash PR(\Gamma, A)}{\Gamma \vdash A} \quad (1\text{-reflect down})$$

We will refer globally to these rules as RR1¹. If $PR(\Gamma, A)$ is a theorem of the extended calculus, then A can be proved from Γ and vice versa. Note that these rules are derivable (provided that a “faithful” axiomatization of provability is included) and therefore no new theorems are generated by virtue of these new rules. Therefore the amalgamation is a conservative extension and provability coincides with provability in the predicate calculus.

In Omega (Attardi and Simi, 1984) the reflection rules are stated as follows (here ‘ s ’ in vp) means that ‘ s ’ can be derived from the set of assumptions ‘ vp ’):

$$\frac{('s' \text{ in } vp)}{vp \vdash s} \quad (\text{reflect down})$$

$$\frac{vp \vdash s}{('s' \text{ in } vp)} \quad (\text{reflect up})$$

That is, if (‘ s ’ in ‘ vp ’) can be derived, then s can be derived from the assumptions vp , and vice versa, if there is a proof of s using a set of assumptions vp , then also (‘ s ’ in ‘ vp ’) can be proved. In the notation of this paper, and making explicit the dependencies, the above rules can be phrased as:

$$\frac{\Gamma_1 \vdash PR(\Gamma_2, A)}{\Gamma_1 \cup \Gamma_2 \vdash A} \quad (2\text{-reflect down})$$

$$\frac{\Gamma_1 \cup \Gamma_2 \vdash A}{\Gamma_1 \vdash PR(\Gamma_2, A)} \quad (2\text{-reflect up})$$

We will refer globally to these rules as RR2.

There is a significant difference between the two formulations of reflection. Rules RR1 are a special case of rules RR2 where Γ_1 is empty. In fact, with (1- reflect down), the only facts that can be reflected down are the theorems of the amalgamated calculus, that is formulas that are derivable using no proper axioms nor assumptions. The assertions about PR made at the meta-level have no influence on the object level. According to rule (2-reflect down) instead, facts that are deduced using a

¹In (Bowen and Kowalski, 1982) the formulation of these rules includes explicitly the axiomatization of the provability relation, denoted as Pr .

set of assumptions can be reflected down in the context of the same assumptions.

Moreover, with (1-reflect up) we can deduce the theorem $PR(\Gamma, A)$ only if there is a valid deduction of A from Γ alone, while with (2-reflect up), if there is a deduction of A from a set of assumptions $\Gamma_1 \cup \Gamma_2$ we can deduce $PR(\Gamma_2, A)$ in the context of Γ_1 ; it is like performing a kind of partial closure on a subset of the assumptions used to deduce A and discharging them, while remaining in the context of the rest of the assumptions.

Note that, with RR2, deduction in the extended calculus is no more classical deduction as in Bowen-Kowalski.

We show that the extension with RR2 is non conservative, by proving a formula across object and meta-level.

A property of deduction that we will use is the following:

$$\text{If } \Gamma \vdash A \text{ then } \Gamma \cup \Gamma_1 \vdash A \quad (\text{monotonicity})$$

which is obvious from the classical definition of derivability.

We will write \vdash_{RR1} and \vdash_{RR2} to indicate explicitly which version of the reflection rules is used in a derivation.

Lemma 1: $\vdash_{RR2} PR(A, B) \Rightarrow (A \Rightarrow B)$

Proof:

$$PR(A, B) \vdash_{RR2} PR(A, B) \quad (\text{definition of } \vdash)$$

$$PR(A, B) \cup A \vdash_{RR2} B \quad (2\text{-reflect down})$$

$$PR(A, B) \vdash_{RR2} (A \Rightarrow B) \quad (\Rightarrow \text{ introduction})$$

$$\vdash_{RR2} PR(A, B) \Rightarrow (A \Rightarrow B) \quad (\Rightarrow \text{ introduction})$$

■

If (2-reflect down) is available, the converse of the implication can also be proved:

Lemma 2: $\vdash_{RR2} (A \Rightarrow B) \Rightarrow PR(A, B)$

Proof:

$$(A \Rightarrow B) \cup A \vdash_{RR2} B \quad (\text{Modus Ponens})$$

$$(A \Rightarrow B) \vdash_{RR2} PR(A, B) \quad (2\text{-reflect up})$$

$$\vdash_{RR2} (A \Rightarrow B) \Rightarrow PR(A, B) \quad (\Rightarrow \text{ introduction})$$

■

If the amalgam is achieved using the inference rules RR1, the proofs of the two lemmas cannot go through. The crucial difference is that in order to apply (1-reflect down) to $PR(A, B)$ this formula must be provable without assumptions, which is not the case here. For a similar

reason (1-reflect up) cannot be used in the proof of Lemma 2.

More general forms of Lemma 1 and Lemma 2 are also derivable:

$$PR(\Gamma, B) \Rightarrow (\wedge\Gamma \Rightarrow B)$$

$$(\wedge\Gamma \Rightarrow B) \Rightarrow PR(\Gamma, B)$$

where the notation $\wedge\Gamma$ should be interpreted as the conjunction of the statements in Γ .

Putting the two lemmas together we get:

$$\text{Theorem 1: } \vdash_{RR2} (\wedge\Gamma \Rightarrow B) \Leftrightarrow PR(\Gamma, B)$$

This result is problematic. In fact undesirable properties of PR will follow, such as:

$$\neg PR(\Gamma, A) \Rightarrow PR(\Gamma, \neg A)$$

$$PR(\Gamma, A \vee B) \Rightarrow PR(\Gamma, A) \vee PR(\Gamma, B)$$

$$(PR(\Gamma, A) \Rightarrow PR(\Gamma, B)) \Rightarrow PR(\Gamma, A \Rightarrow B)$$

$$A \Rightarrow PR(\{\}, A)$$

The last property above was called visibility in (Attardi and Simi, 1984) and was used for a very simple and natural solution to the three wise men puzzle. Visibility is however too strong and sanctions counterintuitive arguments. In the solution of the puzzle we used visibility to model the idea that any agent is able to reproduce the reasoning of another one who shares the same assumptions. There exists a solution to the puzzle, as simple and natural, where (1-reflect up) is used instead of visibility.

More seriously, we can prove that the amalgam using RR2 is inconsistent. For this we refer to the discussion in section 7.

Lemma 1 can be seen as an alternative formulation of (2-reflect down). In fact not only Lemma 1 is derivable using (2-reflect down) but the converse is also true.

Lemma 3: The following axiom and inference rule are equivalent:

$$\vdash_{RR2} PR(\Gamma, A) \Rightarrow (\wedge\Gamma \Rightarrow A) \quad (\text{Lemma 1})$$

$$\frac{\Gamma_1 \vdash PR(\Gamma_2, A)}{\Gamma_1 \cup \Gamma_2 \vdash A} \quad (2\text{-reflect down})$$

Proof: We have to show that (2-reflect down) can be derived from Lemma 1. Assume

$$\Gamma_1 \vdash PR(\Gamma_2, A)$$

then, using Lemma 1 and Modus Ponens

$$\Gamma_1 \vdash (\wedge\Gamma_2 \Rightarrow A)$$

finally, by Modus Ponens,

$$\Gamma_1 \cup \Gamma_2 \vdash A$$

■

As discussed above, Theorem 1 is not derivable with RR1 reflection rules. It turns out however that, for any pair of formulas A and B such that $A \vdash_{RR1} B$, and in particular for those formulas which are provable without assumptions, i.e. theorems of the calculus, the proof can be carried out also with RR1 reflection rules. That is, one can prove at least all the theorems of the form

$$\vdash_{RR1} PR(\Gamma, B) \Leftrightarrow (\wedge\Gamma \Rightarrow B)$$

where $\Gamma \vdash_{RR1} B$. For example:

$$\vdash_{RR1} PR(\{\}, A \Rightarrow A) \Leftrightarrow (A \Rightarrow A)$$

The Bowen-Kowalski amalgamation is consistent if and only if the extended calculus, including the axiomatization of PR but not the reflection rules, is consistent. Even if a full account of consistency was not provided in (Bowen and Kowalski, 1982) the restriction on Theorem 1 that $\Gamma \vdash_{RR1} B$ is sufficient to avoid paradoxes and guarantees consistency. This is a consequence of the results in section 7 where we prove the consistency of a richer logic.

5 A CONSISTENT NON CONSERVATIVE EXTENSION

A consistent non conservative extension can be obtained with the following formulation of the inference rules (RR3):

$$\frac{\Gamma_1 \vdash PR(\Gamma_2, A)}{\Gamma_1 \cup \Gamma_2 \vdash A} \quad (3\text{-reflect down})$$

$$\frac{\Gamma \vdash_{PC} A}{\vdash PR(\Gamma, A)} \quad (3\text{-reflect up})$$

therefore weakening the rule (2-reflect up), which turned out to be problematic while retaining the strong version of reflect down. Note that the precondition in (3-reflect up) is provability in the predicate calculus. More precisely \vdash_{PC} means “derivable using any inference rule except reflect down and reflect up”. This restriction is necessary for consistency (see section 7). Let us call this logic RT .

The extension is still non conservative since one can prove Lemma 1. However Theorem 1 does not hold nor

all its unpleasant consequences. In section 7 we prove that the proposed extension is consistent.

We have several reasons to believe that Lemma 1 is reasonable and desirable. First of all it can be intuitively justified. In case PR is used to represent beliefs, i.e. $PR(\Gamma, A)$ is read as “ A is one of the beliefs of an agent whose assumptions are Γ ”, then Lemma 1 properly models the following property of beliefs: “if an agent holds belief A , then either some of the agent’s assumptions are false or belief A is true”.

In the following section we present an example where the lemma is used to draw conclusions based on an agent’s reliability or lack of reliability.

Lemma 1 can also be regarded as a two argument counterpart of axiom T of modal logics, i.e. $(\Box A \Rightarrow A)$. However, while Lemma 1 is perfectly reasonable when PR is interpreted as belief, axiom T is not, and in fact is usually not included in modal logics for belief.

Finally, we can observe that Lemma 1 looks right when PR is given an intuitionistic reading: in fact if the intuitionistic implication holds then the material implication holds as well but not vice versa.

6 AN EXAMPLE

Using PR , we can provide a formalization of the following puzzle:

The members of the tribe of Voo always speak the truth, while the members of the tribe of Doo always speak the contrary. An explorer meets a Voo and a Doo, and has to figure out which is which.

By denoting with Γ the common knowledge that both Voos and Doos have, we can write:

$$\begin{aligned} \text{Says(Voo, } x) &\Rightarrow PR(\Gamma_V, x) \\ \text{Says(Doo, } x) &\Rightarrow PR(\Gamma_D, \neg x) \end{aligned}$$

where

$$\begin{aligned} \Gamma_V &= \Gamma \cup \{\text{Says(Doo, } x) \Leftrightarrow PR(\Gamma, \neg x)\} \\ \Gamma_D &= \Gamma \cup \{\text{Says(Voo, } x) \Leftrightarrow PR(\Gamma, x)\} \end{aligned}$$

If the explorer asks to one native what the other would answer to a yes/no question, the answer A he would get from a Voo would be false; in fact

$$\begin{aligned} \text{Says(Voo, 'Says(Doo, } A)) &\\ \Rightarrow PR(\Gamma_V, 'Says(Doo, A)) &\\ \Rightarrow PR(\Gamma_V, 'PR(\Gamma, \neg A)) &\\ \Rightarrow PR(\Gamma_V, '\Gamma \Rightarrow \neg A) & \end{aligned}$$

$$\begin{aligned} \Rightarrow (\Gamma_V \Rightarrow (\Gamma \Rightarrow \neg A)) &\\ \Rightarrow (\Gamma \Rightarrow \neg A) & \end{aligned}$$

Similarly the answer of a Doo would be false:

$$\begin{aligned} \text{Says(Doo, 'Says(Voo, A))} &\\ \Rightarrow PR(\Gamma_D, '\neg Says(Voo, A)) &\\ \Rightarrow PR(\Gamma_D, '\neg PR(\Gamma, A)) &\\ \Rightarrow PR(\Gamma_D, 'PR(\Gamma, \neg A)) &\\ \Rightarrow PR(\Gamma_D, '\Gamma \Rightarrow \neg A) &\\ \Rightarrow (\Gamma_D \Rightarrow (\Gamma \Rightarrow \neg A)) &\\ \Rightarrow (\Gamma \Rightarrow \neg A) & \end{aligned}$$

The step in the third line of the second part is justified by the choice of a yes/no question, for whose answer A it is either $PR(\Gamma, A)$ or $PR(\Gamma, \neg A)$. Such is the case for instance for the answer “I am a Voo”. Therefore a suitable question to solve the puzzle could be: “What would your friend answer to the question: «Are you a Voo?»”.

7 THE PROBLEM OF CONSISTENCY

Many authors have discussed the implications of Montague’s negative results about inconsistency of amalgamated theories which have enough technical machinery to represent their own syntax² and strong linking rules. In these theories a self referential formula R can be constructed such that:

$$\text{Lemma 4: } \vdash R \Leftrightarrow PR(\neg R)$$

where $PR(A)$ is an abbreviation for $PR(\{\}, A)$. In order to show the incompleteness of the amalgam, a similar self referential formula was constructed in (Bowen and Kowalski, 1982).

Theorem 2. (Montague) Let T be a theory such that:

- (i) $\vdash PR(A) \Rightarrow A$
- (ii) $\vdash PR(PR(A) \Rightarrow A)$
- (iii) $\vdash PR(A)$, if $\vdash_{PC} A$
- (iv) If $\vdash PR(A \Rightarrow B)$ and $\vdash PR(A)$ then $\vdash PR(B)$
- (v) T includes enough machinery to represent its own syntax

then T is inconsistent.

² This is a minimal requirement that cannot be dispensed with in any reasonable theory (see for example the arguments in (Perlis, 1985)).

The following theorem establishes a tighter correspondence between the two logics.

Theorem 4. *RT and TLN are equivalent*

Proof: *TLN can be translated in RT by means of the following mapping:*

$$p(T('A')) = PR(\{\}, 'p(A)') = PR('p(A)')$$

The translated axioms and inference rules are:

$$IP' \quad PR('A \Rightarrow B') \Rightarrow (PR('A') \Rightarrow PR('B'))$$

$$T' \quad PR('A') \Rightarrow A$$

$$BAR' \quad \forall x PR('A') \Rightarrow PR(' \forall x A')$$

$$LNEC' \quad \frac{\vdash_{PC} A}{\vdash PR('A')}$$

We can observe that **IP'**, **T'** and **LNEC'** are instances of the meta axiom for Modus Ponens, Lemma 1 and reflect up respectively. **BAR'** can be derived using the object level rule for implication elimination and the meta-level axiom for implication introduction. ■

Our goal is to find a consistent amalgamated logic to serve as the basis for a syntactic treatment of modalities. Therefore we are looking for the most powerful consistent logic which is compatible with the interpretation of *PR* as belief.

We have shown that the RR3 formulation of reflection rules results in a logic which is equivalent to the syntactical counterpart of the modal logic *T*. Modal logics for beliefs however usually include a positive introspection axiom (called S4), i.e.

$$PR(' \Gamma, 'A') \Rightarrow PR(' \Gamma, 'PR(\Gamma, A))$$

This is widely accepted as a valid property of belief: "if one believes something, he believes that he believes it". Positive introspection is not however a theorem of *RT*. In fact the truth of *PR(' \Gamma, 'A')* does not imply the existence of a proof; we can derive only a weaker form of positive introspection, namely

$$\text{If } \Gamma \vdash_{PC} A \text{ then } PR(\{\}, 'PR(\Gamma, A))$$

This, at first sight, looks like a disadvantage of our logic over other logics which could, in an intuitionistic style, define *PR* as "derivable". But in such logics, a stronger theorem is derivable, that is³

$$PR(' \Gamma, 'A') \Rightarrow PR(\{\}, 'PR(\Gamma, A))$$

because from the truth of *PR(' \Gamma, 'A')* one would be able to infer that $\Gamma \vdash A$ and therefore the truth of *PR(\{\}, PR(' \Gamma, 'A'))*, by reflecting up twice. This is however not a reasonable principle for belief, since it would amount to the following: "if an agent believes *A* then any other agent believes that the first agent believes *A*".

Davies proves consistent a stronger logic of syntactic modality which includes positive introspection. Based on this result and on the correspondence we have established, it would be safe to add positive introspection as an axiom to logic *RT*.

8 RELATED WORK

Turner (Turner, 1989) shows that a number of choices are available to us for a syntactic treatment of truth and modalities. There are, after all, ways to circumvent, naturally enough, Montague's negative results.

Based on the work by Turner, Davies (Davies, 1990) proposes a first order theory of reasoning agents where truth, belief and knowledge are treated as syntactic predicates and which compares favourably with other similar proposals found in the literature.

Our starting point has been an analysis of reflection rules and a focus on the provability relation. The logic *RT*, which is the result of our investigation, is very similar to one of the logics proposed by Turner and Davies and in fact we could establish a formal correspondence and base our proof of consistency on their results.

We believe that a correct axiomatization of the provability relation can be the starting point also for a formal treatment of truth and other modalities; these, in the work of Turner and Davies, are introduced with separate axiom systems and then integrated.

NProlog ((Gabbay and Reyle, 1985) and (Gabbay, 1985)) introduces hypothetical implications (\rightarrow) in clauses and queries of logic programs. The goal $P ? (A \rightarrow B)$ is taken to be equivalent to the goal $P + A ? B$, that is *B* is a consequence of *P* extended with *A*.

A theoretical motivation for introducing hypothetical implication is to give a definition to the provability predicate, in the context of an amalgamation of language and meta-language in the style of Bowen-Kowalski. In NProlog this is called *NDemo* and is defined according to the following:

$$NDemo('P', 'A') = \text{Suspend and } (\wedge P \rightarrow A) \text{ and Restore}$$

³ This is not the case however for the Bowen-Kowalski's amalgamation.

where P represents a set of clauses (the program), A a formula, and $\wedge P$ the conjunction of all the clauses of P . According to this definition, provability of A from P amounts to trying to prove A after asserting the conjunction of the clauses in P . The use of *Suspend* is "to get rid of the context" which could be used to derive A , and which could produce a positive answer even in the case that A is not derivable from P alone. *Restore* is for restoring the context after the evaluation of the goal. The effect of the right hand side of the definition is therefore equivalent to the existence of a proof (in logic programming) of A from P .

A definition without *Suspend* and *Restore* would have the effect of the (2-reflect up) reflection rule. In fact if $NDemo('P', 'A')$ is defined as $(\wedge P \rightarrow A)$, $NDemo('P', 'A')$ could be proved by asserting $\wedge P$ in the current database and proving A in the context of the extended database.

With the correction of *Suspend* and *Restore* however, *NDemo* succeeds only if the goal of proving A from $\wedge P$ alone succeeds, which is similar to the effect of (1-reflect up).

We have seen that in the Bowen-Kowalski amalgamation, assertions (non logical axioms) concerning PR at the meta-level do not have influence on the object level, due to the restricted form of the rule (1-reflect down). The situation with NProlog is different. According to the above definition, the assertion $NDemo('P', 'A')$ amounts to imposing that the goal $P ? A$ be successful. What if this is not the case?

To conclude, we believe that the only reasonable effect that an assertion $PR(\Gamma, 'A')$ can have on the object level is the one that our Lemma 1 sanctions, that is, the weak semantic constraint (as opposed to "syntactic constraint" concerning derivation) that when Γ is true then A is true.

9 CONCLUSIONS

We argued about the advantages of an amalgamated approach and about the utility of reflection rules which result in a non conservative extension.

We have presented an analysis of reflection rules and discussed the implications of seemingly slight variations in their formulation, specifically in the context of an amalgamated approach.

In the Bowen and Kowalski amalgamation, the interpretation that can be given to PR is intuitionistic: $PR(\Gamma, 'A')$ means $\Gamma \vdash A$.

At the other extreme the Omega approach, which results in PR being equivalent to the classical material implication, is shown to lead to inconsistency.

We have shown that a version of the reflection rules which lays in between the Bowen-Kowalski and the Omega proposal results in the logic RT , which is equivalent to a consistent logic of syntactic modality. In RT , a counterpart of axiom T can be derived which proves useful in some applications.

We think that logic RT is a better candidate to serve as the basis for a theory of viewpoints, where belief is treated as provability in a viewpoint. This aspect will be developed in future work.

Acknowledgements

We are grateful to Daniele Nardi for expressing doubts about the correctness of the proof of the three wise men in (Attardi and Simi, 1984) and to Piero Bonatti for helpful hints. We owe to one of the referees the suggestion that the different formulations of the reflection rules correspond to modal and intuitionistic logics and for bringing to our attention the work on NProlog.

References

- L. C. Aiello, D. Nardi, M. Schaefer (1989) "Reasoning about Knowledge and Ignorance", *Proc. of Int'l Conf. on Fifth Generation Computer Systems*, 618-627.
- G. Attardi, M. Simi (1984) "Metalanguage and Reasoning across Viewpoints", in T. O'Shea (ed.), *ECAI-84: Advances in Artificial Intelligence*, Proc. of 6th European Conference on Artificial Intelligence. Amsterdam, Elsevier Science Publishers.
- K.A. Bowen, R.A. Kowalski (1982) "Amalgamating Language and Metalanguage In Logic Programming", in K. Clark and S. Tarnlund (eds), *Logic Programming*, 153-172. Academic Press.
- N. Davies (1990) "Towards a First Order Theory of Reasoning Agents", *Proc. of 9th European Conference on Artificial Intelligence*, Stockholm.
- D. M. Gabbay, U. Reyle (1985) "N-PROLOG: an Extension of PROLOG with hypothetical implications. I", *The Journal of Logic Programming*, 1; 319-355.
- D. M. Gabbay, (1985) "N-PROLOG: an Extension of PROLOG with hypothetical implications. II. Logical Foundations, and Negation as Failure", *The Journal of Logic Programming*, 4; 251-283.

F. Giunchiglia, L. Serafini, (1990) "Multilanguage First Order Theories of Propositional Attitudes", IRST Technical Report #9001-02, Trento.

F. Giunchiglia, A. Smaill (1989) "Reflection in Constructive and Non-Constructive Automated Reasoning", in Lloyd J. (ed.), *Proc. Workshop on Meta-Programming in Logic Programming*, Cambridge, MA: MIT Press.

K. Konolige (1982) "A First-order Formalization of Knowledge and Action for a Multi-agent Planning System", *Machine Intelligence* 10.

J. McCarthy (1979) "First Order Theories of Individual Concepts and Propositions", *Machine Intelligence* 9, 129–147.

R. Montague (1963) "Syntactical Treatment of Modalities, with Corollaries on Reflexion Principles and Finite Axiomatizability", *Acta Philosoph. Fennica*, (16), 153–167.

R. C. Moore (1977) "Reasoning about Knowledge and Action", *Proc. of IJCAI-77*, 223–227. Cambridge, MA: Morgan Kaufmann.

D. Perlin (1985) "Languages with Self-Reference I: Foundations", *Artificial Intelligence*, (25), 301–322.

D. Perlin (1988) "Languages with Self-Reference II: Knowledge, Belief and Modality", *Artificial Intelligence*, (34), 2, 179–212.

J. R. Shoenfield (1967) *Mathematical Logic*. Reading, MA: Addison Wesley.

R. Turner (1989) *Truth and Modality for Knowledge Representation*. London, Pitman Press.

R. W. Weyhrauch (1980) "Prolegomena to a Theory of Mechanized Formal Reasoning", *Artificial Intelligence*, (13), 133–170.

Nonmonotonic inferences in neural networks

Christian Balkenius
 Cognitive Science
 Department of Philosophy
 University of Lund
 S-223 50 Lund, Sweden
 E-mail: Christian.Balkenius@fil.lu.se,

Peter Gärdenfors
 Cognitive Science
 Department of Philosophy
 University of Lund
 S-223 50 Lund, Sweden
 E-mail: Peter.Gardenfors@fil.lu.se

Abstract

We show that by introducing an appropriate schema concept and exploiting the higher-level features of a resonance function in a neural network it is possible to define a form of nonmonotonic inference relation between the input and the output of the network. This inference relation satisfies some of the most fundamental postulates for nonmonotonic logics. The construction presented in the paper is an example of how symbolic features can emerge from the subsymbolic level of a neural network.

1 INTRODUCTION

Within cognitive science there is a controversy concerning the basic units of cognitive processing. On the one hand there are the so called classical theories (e.g., Fodor and Pylyshyn 1988) where it is argued that the basic units are *symbols* handled by rule-based processes. On the other hand, the connectionist school argues that we should approach cognition at another level and study how *neuronlike* elements interact to produce collectively emerging effects (e.g., Rumelhart et al. 1986).

We believe that it is possible to unify the symbol processing capabilities of the classical theories to the constraint satisfying capabilities of connectionist theories. We want to show that by developing a *high-level* description of the properties of neural networks it is possible to bridge the gap between the symbolic and the subsymbolic levels (see Smolensky 1988). The key concept for this construction will be that of *schemata*. To some extent inspired by earlier schema theories, we will introduce a general schema concept which is appropriate for studying neural networks on levels above the neurons

(see Balkenius 1990). Certain operations on schemata will also be presented.

As an application of the schema concept for neural networks, the aim of this paper is to show that certain activities of such networks can be interpreted as *nonmonotonic inferences*. We shall study these inferences in terms of the general postulates for nonmonotonic logics that have recently been introduced in the literature. It seems that a large class of neural networks will generate so-called *cumulative* nonmonotonic inferences (Makinson 1989).

2 A CONCISE DESCRIPTION OF NEURAL NETWORKS

This section outlines a general description of a class of neural networks. The outline will be used as the starting point for the development of a high-level description based on schemata.

We can define a neural network N as a 4-tuple $\langle S, F, C, G \rangle$. Here S is the space of all possible *states* of the neural network. The dimensionality of S corresponds to the number of parameters used to describe a state of the system. Usually $S = [a, b]^n$, where $[a, b]$ is the working range of each neuron and n is the number of neurons in the system. We will assume that each neuron can take excitatory levels between 0 and 1. This means that a state in S can be described as a vector $x = \langle x_1, \dots, x_n \rangle$ where $0 \leq x_i \leq 1$, for all $1 \leq i \leq n$. The network N is said to be *binary* if $x_i = 0$ or $x_i = 1$ for all i , that is if each neuron can only be in two excitatory levels.

C is the set of possible *configurations* of the network. A configuration $c \in C$ describes for each pair i and j of neurons the connection c_{ij} between i and j . The value of c_{ij} can be positive or negative. When it is positive the connection is *excitatory* and when it is negative it is *inhibitory*. A configuration c is said to be *symmetric* if $c_{ij} = c_{ji}$ for all i and j .

F is a set of *state transition functions* or *activation functions*. For a given configuration $c \in C$, a function $f_c \in F$ describes how the neuron activities spread through that network.

G is a set of *learning functions* that describe how the configurations develop as a result of various inputs to the network. In the sequel, the learning functions will play no significant role.

The two spaces S and C interact by means of the difference equations

$$x(t+1) = f_c(t)(x(t))$$

$$c(t+1) = g_x(t)(c(t))$$

where $s \in S$, $f \in F$, $c \in C$ and $g \in G$.

This gives us two interacting subsystems in a neural network. First, we have the system $\langle S, F \rangle$ that governs the *fast* changes in the network, i.e., the transient neural activity. Then, we have the system $\langle C, G \rangle$ that controls the *slower* changes that correspond to all learning in the system. By changing the behaviour of the functions in the two sets F and G , it is possible to describe a large set of different neural mechanisms. Generally the state transition functions in F have much faster dynamics than the learning functions in G . We will assume that the state in C is fixed while studying the state transitions in S .

Example: In an Interactive Activation network (Rumelhart and McClelland 1986) with four nodes, S is the space $[\min, \max]^4$, C is the space of all 4×4 matrices, and $F = \{f_c(x) = (1-\theta)x + I(c, x) \mid c \in C, x \in S\}$. $I_i(c, x) = c_j x_j (\max - x_i)$ if $c_j x_j > 0$ and $I_i(c, x) = c_j x_j (x_i - \min)$ otherwise. Here the constant $1-\theta$ dampens the activation levels of the neurons and I_i describes the change of the activation level of x_i due to the influence from the other neurons.

The general description of a neural network given here comprises a large class of the systems presented in the literature, for example Hopfield (1984) nets, Boltzmann machines (Ackley et al. 1985), Cohen-Grossberg (1983) models, Interactive Activation models (Rumelhart et al. 1986), the McCulloch-Pitts (1943) model, the BSB model (Anderson et al. 1977), the Harmony networks (Smolensky 1986), and the BAM model (Kosko 1987).

3 SCHEMATA

The basic building block for many theories of cognition is the *schema*. Even though the concept seems to have as many definitions as authors, some common core exists in all of them. We will use the term schema as a collective name of the structures as used by Piaget (1952, 1973), Arbib and Hanson (1987), and Rumelhart et al. (1986). We also want to include concepts usually denoted by other names such as 'frames' (Minsky, 1981 1987), 'scripts' (Schank & Abelson, 1977), etc. Among the various proposals we find some common characteristics of schemata:

- Schemata can be used for representing objects, situations, and actions.

• Schemata have variables. There is thus some way of changing the schema to adapt it to different situations. As a consequence, schemata can be embedded. One schema can have another schema as a part or as an instantiation of a variable.

- Schemata support default assumptions about the environment. They are capable of filling in missing information.

Rumelhart et al. (1986) argue that schemata are sets of 'microfeatures' and show how they emerge from the collective behavior of small neuronlike elements. Each microfeature is represented by a neuronlike unit that interacts with the others by activating or deactivating them. This interpretation of schema lends itself to implementation in constraint satisfying networks (Ackley et al. 1985, Hinton and Sejnowski, 1988, Smolensky 1988, Hopfield 1982, 1984).

Arbib, Conklin and Hill (1987) take a higher level view and make a few additional assumptions about schemata. The framework for their notion is described as being "in the style of the brain." Their idea of schemata is similar to that of Rumelhart's in some respects, but ideas from traditional semantic nets are also used. The environment is represented as an assemblage of schemata, each of which corresponds to an object in the environment.

Minsky's notion of frames is yet another instance of a schema theory at a higher level (Minsky, 1987) with inspiration from both neural theory and semantic nets. If we look at other higher level accounts for schemata, such as Schank's 'scripts', we see more rigid structures. A schema is typically considered to be a set of variables and procedures. When we give the variables values we get an *instantiation* of a schema. This is also the definition of schemata usually used in the field of AI (Charniak & McDermott, 1985).

We want to argue that there is a very simple way of defining the notion of a schema within the theory of neural networks that has the desired properties. The definition we propose is that a schema α corresponds to a vector $\langle \alpha_1, \dots, \alpha_n \rangle$ in the state space S . That a schema α is currently *represented* in a neural network with an activity vector $x = \langle x_1, \dots, x_n \rangle$ means that $x_i \geq \alpha_i$, for all $1 \leq i \leq n$. An equivalent definition is to say that a schema is the *cone* $\alpha = \{z \in S : z_i \geq \alpha_i, \text{ for all } 1 \leq i \leq n\}$ generated by the vector $\langle \alpha_1, \dots, \alpha_n \rangle$ and where α is represented in a neural network with activity vector x iff $x \in \alpha$. There is a natural way of defining a partial order of 'greater informational content' among schemata by putting $\alpha \geq \beta$ iff $\alpha_i \geq \beta_i$ for all $1 \leq i \leq n$. There is a minimal schema in this ordering, namely $0 = \langle 0, \dots, 0 \rangle$ and a maximal element $1 = \langle 1, \dots, 1 \rangle$.

In the light of this definition, let us consider the general desiderata for schemata presented above. Firstly, it is clear that depending on what the activity patterns in a neural

network correspond to, schemata as defined here can be used for representing objects, situations, and actions.

Secondly, if $\alpha \geq \beta$, then β can be considered to be a more general schema than α , and α can thus be seen as an instantiation of the schema β . The part of α not in β is a variable instantiation of the schema β . This implies that all schemata with more information than β can be considered to be an instantiation of β with different variable instantiations. Thus, schemata can have variables even though they do not have any explicit representation of variables. Only the value of the variable is represented and not the variable as such. The index of the instantiation is identified with the added activity vector $\alpha\beta$.

This representation of variables instantiation avoids the combinatorial explosion of the tensor product representation of variable binding suggested by Smolensky (1991) but is weaker in power since it limits the possible embeddings of schemata. For example a schema cannot be recursively embedded into itself. However, the schema concept presented here can very well be used to represent the 'filler' and 'role' structures of Smolensky's constructions, as long as the sets of fillers and roles are disjoint.

Thirdly, the next sections will be devoted to showing that schemata support default assumptions about the environment. The neural network is thus capable of filling in missing information.

There are some elementary operations on schemata that will be of interest when we consider nonmonotonic inferences in a neural network. The first operator is the conjunction $\alpha\beta$ of two schemata $\alpha = \langle \alpha_1, \dots, \alpha_n \rangle$ and $\beta = \langle \beta_1, \dots, \beta_n \rangle$ which is defined as $\langle \gamma_1, \dots, \gamma_n \rangle$, where $\gamma_i = \max(\alpha_i, \beta_i)$ for all i . In terms of cones, $\alpha\beta$ is just the intersection of the cones representing α and β .

If we consider schemata as corresponding to observations in an environment, we can interpret $\alpha\beta$ as the coincidence of two schemata, i. e., the simultaneous observation of two schemata.

Secondly, the complement α^* of a schema $\alpha = \langle \alpha_1, \dots, \alpha_n \rangle$ is defined as $\langle 1-\alpha_1, \dots, 1-\alpha_n \rangle$ (recall that 1 is assumed to be the maximum activation level of the neurons, and 0 the minimum). In general, the complementation operation does not behave like negation since, for example, if $\alpha = \langle 0.5, \dots, 0.5 \rangle$, then $\alpha^* = \alpha$. However, if the neural network is assumed to be binary, that is, if neurons only take activity values 1 or 0, then $*$ will indeed behave as a classical negation on the class of binary-valued schemas.

Furthermore, the interpretation of the complement is different from the classical negation since the activities of the neurons only represent positive information about certain features of the environment. The complement α^* reflects a lack of positive information about α . It can be interpreted as a schema corresponding to the observation of everything but α . As a consequence of this distinction it is pointless to define implication from conjunction and complement. The intuitive reason is that it is impossible

to observe an implication directly. A consequence is that the ordering \geq only reflects greater positive informational content.

However, something similar to classical negation can be constructed in a number of ways. We can let the schema $\langle 0.5, \dots, 0.5 \rangle$ represent a total lack of information. Greater activity will correspond to positive information and lesser activity to negative information. The ordering \geq can be changed to reflect this interpretation if we let $\alpha \geq \beta$ iff $|\alpha_i - 0.5| \geq |\beta_i - 0.5|$ and α_i and β_i both lie on the same side of 0.5, for all i . In this ordering, $\langle 0.5, \dots, 0.5 \rangle$ is the minimal schema and 1 and 0 are both maximal.

Yet another way to construct a negation, which forces us to change the network, is to double the number of nodes in the network and let the new nodes represent the negation of the schema represented in the original nodes. This is achieved by letting the schemata obey the condition $x_i = 1 - x_{n+i}$ (where n is the number of neurons in the original network). It is also possible to impose weaker conditions on the activity of the new nodes to reflect alternative negations.

These alternative ways of defining negation merit further studies. However, in this paper we will, for simplicity, confine ourselves to the first definition given above.

Finally, the disjunction $\alpha \oplus \beta$ of two schemata $\alpha = \langle \alpha_1, \dots, \alpha_n \rangle$ and $\beta = \langle \beta_1, \dots, \beta_n \rangle$ is defined as $\langle \gamma_1, \dots, \gamma_n \rangle$, where $\gamma_i = \min(\alpha_i, \beta_i)$ for all i . The term 'disjunction' is appropriate for this operation only if we consider schemata to represent propositional information. Another interpretation that is more congenial to the standard way of looking at neural networks is to see α and β as two instances of a variable. $\alpha \oplus \beta$ can then be interpreted as the generalization from these two instances to an underlying variable.

It is trivial to verify that the De Morgan laws $\alpha \oplus \beta = (\alpha^* \cdot \beta^*)^*$ and $\alpha \cdot \beta = (\alpha^* \oplus \beta^*)^*$ hold for these operations. The set of all schemata forms a distributive lattice with zero and unit, as is easily shown. It is a boolean algebra if the underlying neural network is binary. In this way, we have already identified something that looks like a propositional structure on the set of vectors representing schemata.

4 RESONANT SCHEMATA

A desirable property of a network that can be seen as performing inferences of some kind is that it, when given a certain input, stabilizes in a state containing the results of the inference. In the theory of neural network such states are called resonant states.

In order to give a precise definition of this notion, consider a neural network $N = \langle S, F, C, G \rangle$. Let us assume that the configuration c is fixed (or changes very slowly) so that we only have to consider one state transition function f_c . For a fixed c in C , let $f_c^0(x) = f_c(x)$ and $f_c^{n+1}(x) = f_c \circ f_c^n(x)$. Then a state y in S is called resonant if it has the following properties:

- (i) $f_c(y) = y$ (equilibrium)
- (ii) If for any $x \in S$ and each $\epsilon > 0$ there exists a $\delta > 0$ such that $|x-y| < \delta$, then $|f_c^n(x)-y| < \epsilon$ when $n \geq 0$ (stability)
- (iii) There exists a δ such that if $|x-y| < \delta$, then $\lim_{n \rightarrow \infty} f_c^n(x) = y$ (asymptotic stability).

Here $|\cdot|$ denotes the standard euclidean metric on the state space S . A neural system N is called *resonant* if for each fixed c in C and each x in S there exists a $n > 0$, that depends only on c and x , such that $f_c^n(x)$ is a resonant state.

If $\lim_{n \rightarrow \infty} f_c^n(x)$ exists, it is denoted by $[x]_c$, and $[\cdot]_c$ is called the *resonance function* for c . It follows from the definitions above that all resonant systems have a resonance function. For a resonant system, we can then define *resonance equivalence* as $x \sim y$ iff $[x]_c = [y]_c$. It follows that \sim is an equivalence relation on S that partitions S into a set of equivalence classes.

A system considered by Cohen and Grossberg (1983) can be written as

$$(1) \quad x_i(t+1) - x_i(t) = a_i(x_i)[b_i(x_i) - \sum_j c_{ij}d_j(x_j)]$$

for a fixed $c \in C$, if $c_{ij} = c_{ji} \geq 0$, $a_i(x_i) \geq 0$ and $d_i'(x_i) \geq 0$. Here a_i which is supposed to be continuous and positive is called the amplification function and b_i the self-signal function. The function d_i which is assumed to be positive and increasing is called the other-signal function and describes how the output from a neuron depends on its activity. It may seem that equation (1) only describes systems where the neurons inhibit each other, but it has been shown that a number of neural models without the restriction $c_{ij} \geq 0$ can be rewritten in the form of (1) by the use of a simple change of coordinates (Grossberg 1989).

Theorem (Cohen and Grossberg 1983): Every trajectory of (1) approaches an equilibrium point.

A consequence of this theorem is that systems that can be described by an equation of the form (1) are resonant systems. Grossberg (1989) shows that the Cohen-Grossberg (1983) model, Hopfield (1984) nets, Boltzmann machines (Ackley et al. 1985), the McCulloch-Pitts (1943) model, the BSB model (Anderson et al. 1977), and the BAM model (Kosko 1987) are resonant systems. Furthermore, it is trivial to show that the Harmony networks (Smolensky 1986) also can be described by an equation in the form of (1) and thus are resonant systems too. A common feature of these types of neural networks is that they are based on *symmetrical* configuration functions C , that is, the connections between two neurons are equal in both directions.

The function $[\cdot]_c$ can be interpreted as filling in *default* assumptions about the environment, so that the schema represented by $[\alpha]_c$ contains information about what the network *expects* to hold when given α as input. Even if α only gives a partial description of, for example, an object,

the neural network is capable of supplying the missing information in attaining the resonant state $[\alpha]_c$.

5 NONMONOTONIC INFERENCES IN A NEURAL NETWORK

We now turn to the problem of providing an *interpretation* of the activities of a neural network that will show it can perform nonmonotonic inferences.

5.1 DEFINITION OF A NONMONOTONIC OPERATION

A first idea for describing the nonmonotonic inferences performed by a neural network N is to say that $[\alpha]_c$ contains the nonmonotonic conclusions to be drawn from α . However, in general we cannot expect the schema α to be included in $[\alpha]_c$, that is, $[\alpha]_c \geq \alpha$ does not always hold. Sometimes a neural network *rejects* parts of the input information – in pictorial terms it does not always believe what it sees.

So if we want α to be included in the resulting resonant state, we have to modify the definition. The most natural solution is to 'clamp' α in the network, that is, to add the *constraint* that the activity levels of all neurons is above α_i , for all i . Formally, we obtain this by first defining a function f_α via the equation $f_\alpha(x) = f(x) \cdot \alpha$ for all $x \in S$. We can then, for any resonant system, introduce the function $[\cdot]_c^\alpha$ for a configuration $c \in C$ as follows:

$$[x]_c^\alpha = \lim_{n \rightarrow \infty} f_\alpha^n(x)$$

This function will result in resonant states for the same neural networks as for the function $[\cdot]_c$ above. The reason is that the difference equation for f_α can be written in the form (1) by replacing all occurrences of x by $x \cdot \alpha$ and then absorbing the conjunction into the functions a , b and d . Since the conditions of the theorem are also fulfilled for the new a , b and d , the Cohen-Grossberg theorem is thus applicable.

Since we are working with a fixed configuration c for a given network, we shall suppress the subscript c in the sequel.

The key idea of this paper is then to define a nonmonotonic inference relation \vdash between schemata in the following way:

$$\alpha \vdash \beta \text{ iff } [\alpha]^\alpha \geq \beta$$

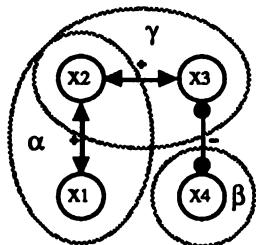
This definition fits very well with the interpretation that nonmonotonic inferences are based on *expectations* as developed in Gärdenfors (1991). Note that α and β in the definition are officially not *propositions* but schemata that are defined in terms of *neural activity vectors* in a neural network. However, in the definition of \vdash they are *treated as propositions*. Thus, in the terminology of Smolensky (1988), we make the transition from the subsymbolic level to the symbolic simply by giving a different *interpretation* of the structure of a neural network. We do this without assuming two different systems as Smolensky does, but

the symbolic level *emerges* from the subsymbolic in one and the same system (cf. Woodfield and Morton 1988). Just like a person may bid at an auction by raising his hand, a neural network may carry out symbolic inferences by performing subsymbolic operations. This kind of double interpretation of an information processing system is also discussed in Gärdenfors (1984).

The symbolic interpretation of neural networks that is based on the schema concept presented here can be applied to *all* neural networks. This is in contrast to some symbol-handling neural networks, like e.g., the μ KLONE system of Derthick (1991), where the propositional structure is a starting point for the construction of the network.

Before turning to an investigation of the general properties of \vdash generated by the definition, we want to illustrate it by showing how it operates for a simple neural network.

Example: The network consists of four neurons with activities x_1, \dots, x_4 . Neurons that interact are connected by lines. Arrows at the ends of the lines indicate that the neurons excite each other; dots indicate that they inhibit each other. If we consider only schemata corresponding to binary activity vectors, it is possible to identify schemata with sets of active neurons. Let three schemata α, β, γ correspond to the following activity vectors $\alpha = <1\ 1\ 0\ 0>$, $\beta = <0\ 0\ 0\ 1>$, $\gamma = <0\ 1\ 1\ 0>$. Assume that x_4 inhibits x_3 more than x_2 excites x_3 . Given α as input the network will activate γ , thus $\alpha \vdash \gamma$. Extending the input to $\alpha \cdot \beta$ causes the network to withdraw γ since the activity x_4 inhibits x_3 . In formal terms $\alpha \cdot \beta \not\vdash \gamma$.



5.1 GENERAL PROPERTIES OF NONMONOTONIC OPERATIONS

One way of characterizing the nonmonotonic inferences generated by a neural network is to study them in terms of the general postulates for nonmonotonic logics that have recently been introduced in the literature (Gabbay 1985, Makinson 1989, 1991, Kraus, Lehmann, and Magidor 1990, Makinson and Gärdenfors 1990, Gärdenfors 1991). We shall present some of these postulates and determine whether they are satisfied for a function $[.]^\alpha$ determined by a transition function in a neural network.

It follows immediately from the definition of $[.]^\alpha$ that \vdash satisfies the property of *Reflexivity*:

$$\alpha \vdash \alpha$$

If we say that a schema β follows logically from α , in symbols $\alpha \vdash \beta$, just when $\alpha \geq \beta$, then it is also trivial to verify that \vdash satisfies *Supraclassicality*:

$$\text{If } \alpha \vdash \beta, \text{ then } \alpha \vdash \beta$$

In words, this property means that immediate consequences of a schema are also nonmonotonic consequences of the schema.

If we turn to the operations on schemata, the following postulate for conjunction is also trivial:

$$\begin{aligned} \text{If } \alpha \vdash \beta \text{ and } \alpha \vdash \gamma, \text{ then } \alpha \vdash \beta \cdot \gamma \\ (\text{And}) \end{aligned}$$

More interesting are the following two properties:

$$\begin{aligned} \text{If } \alpha \vdash \beta \text{ and } \alpha \cdot \beta \vdash \gamma, \text{ then } \alpha \vdash \gamma \\ (\text{Cut}) \end{aligned}$$

$$\begin{aligned} \text{If } \alpha \vdash \beta \text{ and } \alpha \vdash \gamma, \text{ then } \alpha \cdot \beta \vdash \gamma \\ (\text{Cautious Monotony}) \end{aligned}$$

Together Cut and Cautious Monotony are equivalent to each of the following postulates:

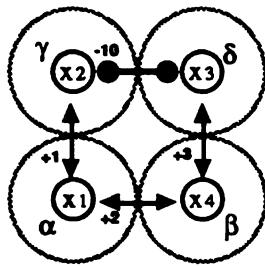
$$\begin{aligned} \text{If } \alpha \vdash \beta \text{ and } \beta \vdash \alpha, \text{ then } \alpha \vdash \gamma \text{ iff } \beta \vdash \gamma \\ (\text{Cumulativity}) \end{aligned}$$

$$\begin{aligned} \text{If } \alpha \vdash \beta \text{ and } \beta \vdash \alpha, \text{ then } \alpha \vdash \gamma \text{ iff } \beta \vdash \gamma \\ (\text{Reciprocity}) \end{aligned}$$

Cumulativity has become an important touchstone for nonmonotonic systems (Gabbay 1985, Makinson 1989, 1991). It is therefore interesting to see that the inference operation defined here seems to satisfy Cumulativity for almost all neural networks where it is defined. However, it is possible to find cases where it is not satisfied:

Counterexample to Reciprocity: The network illustrated below is a simple example of a network that does not satisfy Reciprocity (or Cumulativity). This network is supposed to be *linear* in the sense that the total impact on the activity of a neuron can be written as a *weighted sum* of the excitatory and inhibitory inputs to the neuron, with the c_{ij} 's as weights (represented as +1, +2, +3, and -10 in the figure below).

If we assume that there is an excitatory connection between α and β , it follows that $\alpha \vdash \beta$ and $\beta \vdash \alpha$ since α and β do not receive any inhibitory inputs. Suppose that $\alpha = <1\ 0\ 0\ 0>$ is given as input. Since we have assumed that the inputs to x_3 interact additively, it follows that γ receives a larger input than δ , because of the time delay before δ gets activated. If the inhibitory connection between γ and δ is large, the excitatory input from β can never effect the activity of x_3 . We then have $\alpha \vdash \gamma$ and $\alpha \not\vdash \delta$. If instead $\beta = <0\ 0\ 0\ 1>$ is given as input, the situation is the opposite, and so δ gets excited but not γ , and consequently $\alpha \not\vdash \gamma$ and $\alpha \vdash \delta$. Thus, the network does not satisfy Reciprocity.



A critical factor here seems to be the *linear* summation of inputs that locks x_2 and x_3 to inputs from the outside because the inhibitory connection between them is large.

We have performed extensive computer simulations of networks which obey *shunting* rather than linear summation of excitatory and inhibitory inputs. The simulations suggest that Reciprocity is satisfied in all networks of this kind.

Shunting interaction of inputs is used in many biologically inspired neural network models (e.g., Cohen and Grossberg 1983, Grossberg 1989, Hodgkin 1964, Katz 1966, and Kuffer and Nichols 1976) and is an approximation of the membrane equations of neurons. A simple example of such a network can be described by the following equation:

$$(2) \quad x_i(t+1) = x_i(t) + \delta(1-x_i(t))\sum_j d(x_i(t))c_{ji}^+ + \delta x_i(t)\sum_j d(x_i(t))c_{ji}^-$$

Here δ is a small constant, c_{ij}^+ and c_{ij}^- are matrices with all $c_{ij}^+ = c_{ji}^+ \geq 0$, and all $c_{ij}^- = c_{ji}^- \leq 0$; $d(x) \geq 0$ and $d'(x) > 0$. The positive inputs to neuron x_i are shunted by the term $(1-x_i(t))$ and the negative inputs by $x_i(t)$. As a consequence, the situation where one input locks another of opposite sign cannot occur, in contrast to the linear case above. In other words, a change of input, that is a change in $\sum_j d(x_i(t))c_{ji}^+$ or $\sum_j d(x_i(t))c_{ji}^-$, will always change the equilibrium of x_i .

We believe that the fact that one input never locks another of opposite sign is the reason why all the simulated shunting networks satisfy Reciprocity.

Simulation example: As an example of a simulation we start from the network illustrated above. The matrix for the values of c_{ij}^+ and c_{ij}^- in equation (2) is as follows:

Table 1: Connection Matrix

0	1	0	2
1	0	-10	0
0	-10	0	3
2	0	3	0

The function d in the equation was chosen as $d(x_i) = (\max(x_i - \theta, 0))^2 / ((\max(x_i - \theta, 0))^2 + 0.2)$, where θ is an

output threshold which was chosen to be 0.1. Finally, the constant δ was set to 0.005. (The values of the constants are not crucial for the outcome of the simulation).

In the simulation, the atomic schemata correspond to the activities x_1, \dots, x_4 of the single neurons. For these neurons there are 16 different combinations of *binary* input schemata. In the table below these input schemata are indicated by bold face values. This represents clamped activities of the neurons according to the definition of the function $[x]_C^\alpha$.

The vectors in the table describe the resonant states induced by the different input vectors. Except for the limiting case 0, there are four different resonant states for this network. It is easy to check that the nonmonotonic inference relation generated by this set of resonant states satisfies Reciprocity. For example, in case 2 we have $\gamma \vdash \alpha$ and in case 3 we have $\gamma \cdot \alpha \vdash \beta$ and in accordance with Reciprocity (or Cut), we also find in case 2 that $\alpha \vdash \beta$.

Table 2: Resonant States

run	x1	x2	x3	x4
0:	0.00	0.00	0.00	0.00
1:	1.00	0.12	0.71	1.00
2:	1.00	1.00	0.23	1.00
3:	1.00	1.00	0.23	1.00
4:	1.00	0.09	1.00	1.00
5:	1.00	0.09	1.00	1.00
6:	1.00	1.00	1.00	1.00
7:	1.00	1.00	1.00	1.00
8:	1.00	0.12	0.71	1.00
9:	1.00	0.12	0.71	1.00
10:	1.00	1.00	0.23	1.00
11:	1.00	1.00	0.23	1.00
12:	1.00	0.09	1.00	1.00
13:	1.00	0.09	1.00	1.00
14:	1.00	1.00	1.00	1.00
15:	1.00	1.00	1.00	1.00

For the disjunction operation it does not seem possible to show that any genuinely new postulates for nonmonotonic inferences are fulfilled. The following special form of transitivity is a consequence of Cumulativity (cf. Kraus, Lehmann, and Magidor (1990), p. 179):

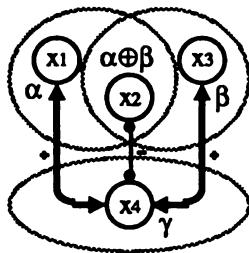
If $\alpha \oplus \beta \vdash \alpha$ and $\alpha \vdash \gamma$, then $\alpha \oplus \beta \vdash \gamma$

This principle is thus satisfied whenever Cumulativity is.

The general form of Transitivity, i.e., if $\alpha \vdash \beta$ and $\beta \vdash \gamma$, then $\alpha \vdash \gamma$, is not valid for all α , β , and γ , as can be shown by the first example above. Nor is the following principle generally valid:

If $\alpha \vdash \gamma$ and $\beta \vdash \gamma$, then $\alpha \oplus \beta \vdash \gamma$
(Distribution)

Counterexample to Distribution: The following network is a simple counterexample: x_1 excites x_4 more than x_2 inhibits x_4 . The same is true for x_3 and x_2 . Giving $\alpha = <1\ 0\ 0>$ or $\beta = <0\ 1\ 1\ 0>$ as input activates x_4 , thus $\alpha \vdash \gamma$ and $\beta \vdash \gamma$. The neuron x_2 which represents schema $\alpha \oplus \beta$ on the other hand has only inhibitory connections to x_4 . As a consequence $\alpha \oplus \beta \not\vdash \gamma$.



6 CONCLUSION

We have shown that by introducing an appropriate schema concept and exploiting the higher-level features of a resonance function in a neural network it is possible to define a form of nonmonotonic inference relation. It has also been established that this inference relation satisfies some of the most fundamental postulates for nonmonotonic logics. The construction presented in this paper is an example of how symbolic features can emerge from the subsymbolic level of a neural network.

Acknowledgements

Research for this article has been supported by the Swedish Council for Research in the Humanities and Social Sciences. We want to thank the referees and the participants of the Cognitive Science seminar in Lund for helpful comments.

References

- Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985): "A learning algorithm for Boltzmann machines," *Cognitive Science* 9, 147-169.
- Anderson, J. A., Silverstein, J. W., Ritz, S. R. & Jones, R. S. (1977): "Distinctive features, categorial perception, and probability learning: Some applications of a neural model," *Psychological Review* 84, 413-451.
- Arbib, M. A., Conklin, E. J., and Hill, J. C. (1987): *From Schema Theory to Language*. New York: Oxford University Press.
- Arbib, M. A. and Hanson, A. R. (1987): *Vision, Brain, and Cooperative Computation*. Cambridge, MA: MIT Press.
- Balkenius, C. (1990): "Neural mechanisms for self-organization of emergent schemata, dynamical schema processing, and semantic constraint satisfaction," manuscript, Cognitive Science, Department of Philosophy, Lund University.
- Charniak, E., and McDermott, D. (1985): *Introduction to Artificial Intelligence*. New York: Addison-Wesley.
- Cohen, M. A., and Grossberg, S. (1983): "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13, 815-826.
- Derthick, M. (1990): "Mundane reasoning by settling on a plausible model," *Artificial Intelligence* 46, 107-157.
- Fodor, J. and Pylyshyn, Z. (1988): "Connectionism and cognitive architecture: A Critical Analysis," in S. Pinker & J. Mehler (eds.) *Connections and Symbols*. Cambridge, MA: MIT Press.
- Gabbay, D. (1985): "Theoretical foundations for non-monotonic reasoning in expert systems," in *Logic and Models of Concurrent Systems*, K. Apt ed., Berlin: Springer-Verlag.
- Gärdenfors P. (1984): "The dynamics of belief as a basis for logic," *British Journal for the Philosophy of Science* 35, 1-10.
- Gärdenfors P. (1991): "Nonmonotonic inferences based on expectations: A preliminary report," these *Proceedings*.
- Grossberg, S. (1978): "A theory of human memory: self-organization and performance of sensory-motor codes, maps, and plans," *Progress in Theoretical Biology*, Vol. 5, Academic Press, 233-374.
- Grossberg, S. (1989): "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures," in *Neural Networks*, Vol. 1, pp. 17-66.
- Hinton, G. E. and Sejnowski, T. J. (1988): "Learning and relearning in Boltzmann machines," in Rumelhart, D.E., *Parallel Distributed Processing*, Vol 1, 282-317, Cambridge, MA: MIT Press.
- Hodgkin, A. L. (1964): *The conduction of the nervous impulse*, Liverpool: Liverpool University Press.
- Hopfield, J. J. (1982): "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences* 79.
- Hopfield, J. J. (1984): "Neurons with graded response have collective computational properties like those of two-state

- neurons." *Proceedings of the National Academy of Sciences* 81.
- Katz, B. (1966): *Nerve, muscle, and synapse*, New York: McGraw-Hill.
- Kosko, B. (1987): "Constructing an associative memory," *BYTE Magazine* 12, 137-144.
- Kraus, S., Lehmann, D., and Magidor, M. (1990): "Nonmonotonic reasoning, preferential models and cumulative logics," *Artificial Intelligence* 44, 167-207.
- Kuffer S. W. and Nicholls, J. G. (1976): *From neuron to brain*, Sunderland, MA: Sinauer Associates, 1976.
- Makinson, D. (1989): "General theory of cumulative inference," in *Non-Monotonic Reasoning*, M. Reinfrank, J. de Kleer, M.L. Ginsberg, and E. Sandewall, eds., Berlin: Springer Verlag, Lecture Notes on Artificial Intelligence n° 346.
- Makinson, D. (1991): "General patterns in nonmonotonic reasoning," to appear as Chapter 2 of *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume II: Non-Monotonic and Uncertain Reasoning*. Oxford: Oxford University Press.
- Makinson, D. and Gärdenfors, P. (1990): "Relations between the logic of theory change and nonmonotonic logic," to appear in *Proceedings of the Konstanz Workshop on Theory Change*, A. Fuhrmann and M. Morreau, eds., Berlin: Springer Verlag.
- McCulloch, W. S. and Pitts, W. (1943): "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics* 5, 115-133.
- Minsky, M. (1981): "A framework for representing knowledge," in J. Haugeland (Ed.), *Mind Design*, Cambridge, MA: The MIT Press.
- Minsky, M. (1987): *The Society of Mind*. London: Heinemann.
- Piaget, J. (1952): *The Origins of Intelligence in Children*. New York: International University Press.
- Piaget, J. and Inhelder, B. (1973): *Memory and Intelligence*. London: Routledge & Kegan Paul.
- Rumelhart, D. E. (1988): *Parallel distributed processing*, Vols I and II. Cambridge, MA: MIT Press.
- Rumelhart, D. E., Smolensky, P., McClelland, J. L. and Hinton, G. E (1986): "Schemata and sequential thought processes in PDP models," in Rumelhart, D.E., *Parallel Distributed Processing, Vol 2*, 7-57, Cambridge, MA: MIT Press.
- Schank, R. and Abelson, R. P. (1977): *Scripts, plans, goals, and understanding*. Hillsdale, NJ: Lawrence Erlbaum.
- Smolensky, P. (1986): "Information processing in dynamical systems: foundations of harmony theory," in Rumelhart, D.E., *Parallel Distributed Processing*, Vol 1, 194-281, Cambridge, MA: MIT Press.
- Smolensky, P. (1988): "On the proper treatment of connectionism," *Behavioral and Brain Sciences* 11, 1-23.
- Smolensky, P. (1991): "Tensor product variable binding and the representation of symbolic structures in connectionist systems," *Artificial Intelligence* 46, 159-216.
- Woodfield, A. and Morton, A. (1988): "The reality of the symbolic and subsymbolic systems," *Behavioral and Brain Sciences* 11, 58.

Meta-Reasoning in Executable Temporal Logic

Howard Barringer **Michael Fisher**
 Department of Computer Science,
 University of Manchester,
 Oxford Road,
 Manchester M13 9PL, UK.
 {howard,michael}@cs.man.ac.uk

Dov Gabbay **Anthony Hunter**
 Department of Computing,
 Imperial College of Science, Technology and Medicine,
 180 Queen's Gate,
 London SW7 2BZ, UK.
 {dmg,abh}@doc.ic.ac.uk

Abstract

Temporal logic can be used as a programming language. If temporal formulae are represented in the form of an implication where the antecedent refers to the past, and the consequent refers to the present and the future, then formulae that have their antecedent satisfied can be satisfied by considering the consequent as an imperative to be obeyed. Such a language becomes a natural alternative to programming languages, such as Prolog, for temporal reasoning. Here, the approach is extended to include an executable meta-language. This is of importance in developing interpretation, debugging, loop-checking, and simulation tools, and provides a representation for providing control knowledge in the execution of planning and scheduling programs.

1 Executable Temporal Logic

The traditional view of temporal logic is that it represents declarative statements about the world, or about possible worlds over time. Such statements relate the truth of propositions in the past, present and future, and so have been found useful in the representation of time-dependent systems (Pnueli, 1977; Allen and Hayes, 1985). Using this declarative approach, time is viewed from the *outside* — it is seen in its entirety. An alternative view of temporal logic is to consider it in terms of a declarative past and an *imperative* present and future, based on the intuition that a statement about the future can be imperative, initiating steps of action to ensure it becomes true. Using this view, time is seen from the *inside* — we are at a moment in time with the past behind us and the future yet to happen, but with ‘instructions’ about what to do in the future. In such an approach, the future must be actually made to happen, rather than just stating its properties. It is this second approach that we follow.

Temporal specifications can be given, naturally, as a collection of rules of the form

ANTECEDENT ABOUT THE PAST \Rightarrow
CONSEQUENT ABOUT THE PRESENT AND FUTURE

For example, if the flow of time is represented as discrete linear order, e.g. the natural numbers, then atomic propositions are true or false at points on that line. Executable temporal logic statements can then be executed at a point X on that line by reviewing which atomic propositions were true in the past, i.e. at points less than X , and then taking steps to ensure the satisfaction of the specification. If any of the necessary steps include an action in the present then, to ensure satisfaction of the specification, that action must be undertaken now. If any of the necessary steps include an action in the future then this instruction must be passed on to the future.

Executable temporal logics of this form that have been developed include USF (Gabbay, 1989) and METATEM (Barringer et al., 1989; Fisher, 1989). Such logics are based on *US logic*¹ and incorporate a notion of interaction with the environment of the program. Predicates and propositions are partitioned into those controlled by the *component*, i.e. the program, and those controlled by the *environment*, i.e. the context of the program. Thus, at certain times in the execution of such a temporal formula, the environment may need to be consulted in order to provide the values of those atoms controlled by the environment.

If specifications are given in the above rule-based form, then, at any moment in time, the past can be checked and the future-time consequents of all the rules whose antecedents are satisfied can be collected together. These constraints on current and future behaviour can

¹ US logic is a temporal logic with until (‘ U ’) and since (‘ S ’) as its only temporal operators.

be rewritten as a formula of the form

$$\bigvee_i (\bigwedge_j l_{ij} \wedge \bigcirc \varphi_i)$$

where each l_{ij} is a literal and each φ_i is a present and future-time temporal formula. Thus, at every moment in time, the executing agent must endeavour to satisfy the above formula. It can do this by making any one of the disjuncts true. Choosing which disjunct to make true is a subtle problem, and this choice will depend on several factors, such as any commitments to satisfy other clauses, possible future deadlocks, and the environment at that point in time. There may be more than one valid choice at each time, and the actual choice may vary over time.

All formulae of propositional and first-order US-logic can be rewritten into the above executable form (Gabbay, 1989), although the executable form is not, in general, unique, i.e. a specification can be written into more than one equivalent form. For the propositional case, the execution is decidable, i.e. if a US-logic program is satisfiable, then a model for the program can be found through execution. First-order US-logic is, in general, not decidable, hence for applications where it is crucial to find a model if the program is satisfiable, we can either adopt a decidable subset of first-order US-logic, or harness heuristics that ameliorate the problem.

A METATEM execution mechanism has been defined for temporal logic programs that returns a model of the program if the program is satisfiable (Barringer et al., 1989). This result is conditional on the program comprising only component propositions. If environment atoms are introduced into a program, then branching time logic may be required in order to represent all possible environments (Clarke and Emerson, 1982; Pnueli and Rosner, 1989).

Executable temporal logic provides a natural alternative to programming languages, such as Prolog, for temporal reasoning. A temporal logic program given in our executable form provides a clear link between a declarative past and an imperative present and future, and is being developed for applications in planning, scheduling and process control. Though an executable temporal logic that incorporates the full generality of temporal logics can be inefficient, restricted temporal languages have been shown to be useful in many areas (Moszkowski, 1986; Fujita et al., 1986; Hattori et al., 1986; Caspi et al., 1987; Abadi, 1987).

The executable temporal logic considered in this report is one of the family of METATEM languages (Barringer et al., 1989). Other members of this family have been implemented (Fisher, 1989; Owens, 1990) and such languages have been used successfully in large case studies, for example the development of a PAYE ('Pay As You Earn') tax system (Torsun and Manning, 1990).

In this paper, we focus on an executable meta-language. We indicate why a meta-language is of interest, outline the logical basis of a meta-language for executable temporal logic, and consider applications in knowledge representation and reasoning.

2 Why Use A Meta-Language?

In (Barringer et al., 1989), we introduced propositional and first-order temporal logics (PML and FML) as instances of METATEM. We suggested that a rather different logic is required to carry out meta-level reasoning within METATEM. By meta-level reasoning, we mean the ability, in one language (the *meta-language*), to name and reason about statements from another language (the *object-language*). In the case of METATEM we require one (logical) language to be both the meta-language and the object-language, i.e., a language that can act as its own meta-language. Of course, this is not new in its own right, for example in Prolog meta-level reasoning about clauses can be carried out within the same language (Hill and Lloyd, 1988; van Harmelen, 1988; Nilsson and Małuszyński, 1990).

Using a first-order temporal logic, such as FML (Barringer et al., 1989), a wide variety of temporal specifications can be constructed and represented as METATEM programs. However, from a computational point of view greater expressive power is needed, as there is a requirement to be able to interact with, and to influence, a running METATEM process through meta-level manipulation.

Many useful programming tools are meta-level tools, for example, compilers, debuggers, partial evaluators, etc. Also, in declarative languages, many techniques for improving efficiency can be implemented by meta-level manipulation, for example program transformation techniques such as folding and unfolding. From a logical point of view, we can also see the benefit of meta-level statements as they give us the ability, for example, to change the logical basis of the interpreter. A particularly useful example of this is the ability to represent control knowledge to restrict or prioritize the possible execution paths (as for example in simulation or scheduling tasks). Thus, we require a logic that has some way of representing its own formulae in its domain of discourse, in particular, a logic for METATEM in which METATEM program rules can be represented.

There are several standard methods for developing meta-languages. One of the most widespread is the use of *quoting*. This technique, proposed by Frege and further developed by many others (Perlis, 1985), gives a way of producing a term in the logic that represents or 'names' a formula in the logic. Thus, if we have the formula $p(x) \wedge q(x)$, a new term representing the formula is given by " $p(x) \wedge q(x)$ ". As predicates are applied to terms of the language, they can obvi-

ously be applied to these ‘naming terms’, for example $r("p(x) \wedge q(x)")$.

Thus, we can write the following type of formula in a temporal logic incorporating such a quoting mechanism²,

$$\bullet p("x \wedge y") \Rightarrow q("x") \wedge r("y").$$

In such a way a logic capable of meta-level reasoning can be developed. In the case of METATEM, such a logic, called FML*, based on quoting has been developed (Fisher, 1990). The questions of the reduction of ‘naming’ terms to their constituent parts, the quoting of variables, and the scope of quantification are discussed elsewhere (Perlis, 1985; Fisher, 1990). Furthermore, the FML* approach corresponds to the *ground representation* described in (Hill and Lloyd, 1988).

In this report we will present an alternative approach which unifies object-level and meta-level into a single language without using explicit quoting. In our logic, called Meta-METATEM Logic, the set of terms is partitioned into object-level terms and meta-level terms. Object-level terms are the basic terms of the language; meta-level terms are terms that represent (or name) formulae in the language.

Though a technical definition of Meta-METATEM Logic (MML) is given in appendix A, a brief outline of the principles underlying MML will be given in the next section.

3 Meta-METATEM Logic (MML)

MML is a first-order temporal logic, based on a linear, discrete model of time. MML contains the standard temporal operators, such as ‘sometime in the future’ (\Diamond), ‘always in the future’ (\Box) and ‘at the next moment in time’ (\bigcirc), along with their past-time counterparts, and uses a non-rigid form of quantification. A semantics for MML based on possible-worlds is given in appendix A and is outlined below.

In MML the domain over which terms range has been extended to incorporate the *names* of object-level formulae. As we are using MML as its own meta-language, this domain includes the names of *all* MML formulae. We use a *typed representation* (Hill and Lloyd, 1988) for the meta-language, and partition variables into two sorts, o , representing object-level terms, and μ , representing meta-level terms. The terms of MML can thus be partitioned into two sets:

\mathcal{L}_{ot} — the set of *object terms*.

This contains those finite terms that are either variables of sort o or constants³, or are generated

²Here, we use a discrete model of time and use ‘ \bullet ’ as the operator representing ‘at the last moment in time’.

³All constants are object-level terms.

by application of function symbols to other elements of \mathcal{L}_{ot} . Such terms have sort o and can be considered as the basic (object-level) terms of the language.

$\mathcal{L}_{\mu t}$ — the set of *meta-level terms*.

This set contains variables of sort μ , all the finite formulae constructed from elements of $\mathcal{L}_{\mu t}$, and predicates applied to terms in $\mathcal{L}_{\mu t} \cup \mathcal{L}_{ot}$. Such terms have sort μ and can be considered as the names of all the formulae in the language (i.e. the meta-level terms).

Thus, $\mathcal{L}_{\mu t}$ is effectively the set of well-formed formulae (wff) of the both the object and meta-levels, together with meta-level variables.

As an example, consider the term $p(f(a, b, c))$ where p is a predicate, f is a function and a , b and c are constants. Here, $f(a, b, c)$ is a term in \mathcal{L}_{ot} , yet when a predicate is applied to it, it becomes a term in $\mathcal{L}_{\mu t}$, i.e. $p(f(a, b, c))$ is a term in $\mathcal{L}_{\mu t}$.

Note that when we define WFF_m (well-formed formulae of MML) then for every formula in WFF_m , there is a corresponding term in $\mathcal{L}_{\mu t}$. For example, $p(f(a, b, c))$ is in WFF_m , but it is also a term in $\mathcal{L}_{\mu t}$. This duplication is essential as the elements of $\mathcal{L}_{\mu t}$ act as ‘names’ for the formulae in WFF_m . This will not cause any technical problems since elements of $\mathcal{L}_{\mu t}$ are only used as names for formulae. The construction of $\mathcal{L}_{\mu t}$ is a recursive (and long) process.

We now turn to the semantics of MML. Recall that the domain over which terms range is not simply the domain of \mathcal{L}_{ot} , it is also the domain of $\mathcal{L}_{\mu t}$, the names of formulae in MML. Note that, as we are using a temporal logic with non-rigid interpretations, we require that interpretations be indexed by the state (or world) at which the symbols are to be evaluated. As all variables have a sort, we can simply bind them to elements of the appropriate domain. In classical logics, the interpretation of a predicate symbol usually gives a signature of the form

$$\mathcal{D}^n \rightarrow \{T, F\}.$$

Thus, depending on the values of its arguments, the predicate is interpreted as either T or F . In MML, we not only have to index this interpretation with the state at which the predicate is to be evaluated, but there is also the possibility of the predicates’ arguments being in $\mathcal{L}_{\mu t}$ (as well as \mathcal{L}_{ot}). Thus, we interpret predicates using

$$(\mathcal{S} \times (\mathcal{D} \cup \mathcal{T})^n) \rightarrow \{T, F\}$$

where \mathcal{S} is the set of states (possible-worlds) and \mathcal{T} is the domain associated with $\mathcal{L}_{\mu t}$, i.e. the ground terms in $\mathcal{L}_{\mu t}$.

One peculiarity of this approach is that if we have a variable assignment that maps x to φ , where x is of

sort μ and $\varphi \in \mathcal{L}_{\mu}$, then x is replaced by φ throughout the formula. Thus,

$$p(x) \wedge q(x, r(x))$$

becomes

$$p(\varphi) \wedge q(\varphi, r(\varphi)).$$

In summary, the main difference between MML and a standard first-order temporal logic (apart from the scoping of variable quantification, as described above) is that the interpretation of predicate symbols is not just a map from elements of the base domain to $\{T, F\}$, but is a map from elements of the base domain and the domain of names of formulae (T) to $\{T, F\}$. (This is why, in the example given in appendix A.3, the interpretation function for predicate symbols maps formulae, as well as terms, to $\{T, F\}$.)

One further complication is that as we are dealing with a temporal logic, π_p is also parameterised by the state at which it is evaluated, i.e.,

$$\pi_p: (S \times \mathcal{L}_p) \rightarrow ((\mathcal{D} \cup T)^n \rightarrow \{T, F\}).$$

For an example of the use of this semantics, see appendix A.3.

4 Applications

We now give some examples of the use of meta-level representation in METATEM. The notation used will be that of MML and the underlying execution mechanism will be that outlined in section 1.

Some of the obvious applications of meta-level reasoning will be described, such as control of execution and the construction of specialized meta-interpreters, such as those for exploring possible futures. Apart from the applications described here, other aspects of meta-programming that are being explored elsewhere include the implementation of concurrent and object-oriented versions of METATEM, dynamic learning of temporal rules, and the development of compilers, debuggers and partial evaluators.

4.1 Building an interpreter for PML

To show how the meta-level techniques described above can be used in practice, we give, as an example, an interpreter for propositional METATEM (PML) written in MML. Note that the execution mechanism assumes that the MML interpreter contains a forward chaining inference mechanism, not only for temporal rules, but also for solving state-based constraints and that the variables within the rules are universally quantified.

To show the split of rules between state-based and temporal constraints, we will present these two categories separately. However, in a ‘real’ implementation, there may be no need for such a split.

First, we give the rules for a naive PML interpreter; later we add more sophisticated features. The *State Rules* for the interpreter (which is encapsulated within the `execute()` predicate⁴) are given in figure 1.

<code>execute(◻φ)</code>	\Leftrightarrow	<code>execute(φ ∧ ◇ ◻φ)</code>
<code>execute(◊φ)</code>	\Leftrightarrow	<code>execute(φ ∨ (¬φ ∧ ◇ ◊φ))</code>
<code>execute(¬p)</code>	\Leftrightarrow	<code>¬execute(p)</code>
<code>execute(φ ∧ ψ)</code>	\Leftrightarrow	<code>execute(φ) ∧ execute(ψ)</code>
<code>execute(φ ⇒ ψ)</code>	\Leftrightarrow	<code>execute(φ) ⇒ execute(ψ)</code>
<code>execute(φ ∨ ψ)</code>	\Leftrightarrow	<code>execute(φ) ∨ execute(ψ)</code>
<code>(prop(p) ∧ env(p))</code>	\Rightarrow	<code>(execute(p) ⇔ read(p))</code>
<code>(prop(p) ∧ comp(p))</code>	\Rightarrow	<code>(execute(p) ⇔ do(p))</code>

Figure 1: State Rules for a PML Interpreter

The basic *Temporal Rule* is

$$\bullet \text{execute}(\bigcirc \varphi) \Rightarrow \text{execute}(\varphi).$$

The idea behind these rules is that they are applied globally, with the translation rules being used to execute any current-state constraints, and the temporal rule being used to move from one state to the next. The predicates `prop()`, `env()`, `comp()`, `read()`, and `do()` are basic system predicates supplied by the base interpreter. `prop()` is true if its argument is a proposition, `env()` is true if its argument is controlled by the environment, `comp()` is true if its argument is controlled by this component, `read()` is true if its argument has been set to `true` by the environment, and `do()` is true when its argument is forced to be `true` by this component.

As an example of the execution of this meta-interpreter, consider the execution of the PML formula $r \Rightarrow \bigcirc \diamond a$, where r is an environment proposition and a is a component proposition. Thus, we wish to make `execute(r ⇒ ◇ ◊ a)` true. The translation rules above are used to rewrite this formula as follows

* `execute(r ⇒ ◇ ◊ a)` becomes

$$\text{execute}(r) \Rightarrow \text{execute}(\bigcirc \diamond a)$$

using the rule for \Rightarrow ,

* the object-level system interprets

$$\text{execute}(r) \Rightarrow \text{execute}(\bigcirc \diamond a)$$

as

‘if `execute(r)` then `execute(◇ ◊ a)`’,

⁴Note that when we use equivalences such as $A \Leftrightarrow B$, where A is a past-time formula, we use this as an abbreviation for the rules $A \Rightarrow B$ and $\neg A \Rightarrow \neg B$. If we use $A \Leftrightarrow B$, where neither A nor B are past-time formulae, we use this as an abbreviation for $\bullet \text{true} \Rightarrow ((A \wedge B) \vee (\neg A \wedge \neg B))$.

- * as r is an environment proposition, $\text{execute}(r)$ becomes $\text{read}(r)$,
- * if $\text{read}(r)$ is false, then the original formula is trivially satisfied, but if $\text{read}(r)$ is true, then $\text{execute}(\bigcirc\Diamond a)$ is not expanded further until the object-level interpreter moves to the next moment in time where the temporal rule is invoked, generating $\text{execute}(\Diamond a)$.

And so on. The computation continues forever unless the formulae being executed are contradictory. Note that the rewriting performed in every state will always terminate as we only use finite formulae and that once this rewriting has terminated, the object-level interpreter will move on to the next moment in time.

Now, notice that, given the rule for executing $\Diamond\varphi$ above, it is perfectly acceptable for the interpreter to continually execute $\neg\varphi$. We can try to avoid this behaviour and force the meta-interpreter to execute eventualities by adding the temporal rule

$$\bullet (\text{execute}(\Diamond\varphi) \wedge \neg\text{execute}(\varphi)) \Rightarrow \Diamond\text{execute}(\varphi)$$

Here the responsibility for executing the eventuality is passed on to the interpreter that is actually executing the meta-interpreter (i.e., the object-level interpreter). However, if the object-level interpreter does not guarantee to execute the eventuality, we are left with the same problem. A solution to this problem is to ensure that some form of loop-checking is present.

4.1.1 Loop Checking

A possible development of the interpreter is to incorporate heuristics for identifying potential looping in an execution of an object-level specification. For example, if we are operating without an environment, we can consider that if the set of commitments remains constant over a period of time, and that this set of commitments includes eventuality commitments then we are failing to satisfy these eventualities. So if we take the simple interpreter, defined above, we can amend the definition of $\text{execute}(\varphi \vee \psi)$ so that if one of the disjuncts represents an eventuality that has been outstanding for a sufficiently long time, then that disjunct will be chosen. The extra rule are given in figure 2.

$$\begin{aligned} & \neg(\text{execute}(\varphi \vee \psi) \wedge \text{loop}(\varphi) \wedge \text{loop}(\psi)) \\ & (\text{execute}(\varphi \vee \psi) \wedge \neg\text{loop}(\varphi) \wedge \text{loop}(\psi)) \quad \Rightarrow \text{execute}(\psi) \\ & (\text{execute}(\varphi \vee \psi) \wedge \text{loop}(\varphi) \wedge \neg\text{loop}(\psi)) \quad \Rightarrow \text{execute}(\varphi) \\ & (\text{execute}(\varphi \vee \psi) \wedge \neg\text{loop}(\varphi) \wedge \neg\text{loop}(\psi)) \quad \Rightarrow \text{execute}(\varphi) \vee \text{execute}(\psi) \end{aligned}$$

Figure 2: Extra Rules for Loop-Checking

Here, the predicate $\text{loop}()$ is defined so that it is true if the formula given as its argument has appeared in the set of commitments for the last n states (i.e. its argument has been an argument to $\text{execute}()$ for the last n states, but is an eventuality that has not been satisfied in this time). If the specification is being executed without interacting with its environment, the value n can be extracted from the specification itself as the upper bound on the size of the automaton representing models of the specification (Vardi and Wolper, 1986)⁵. If interaction with the environment occurs, the value n cannot be deduced directly and must either be defined explicitly or derived through the use of heuristics.

4.2 Planning Linear Execution using a Branching Interpreter

Another example of the use of meta-level techniques in constraining the execution of METATEM is the use of a ‘branching interpreter’ to explore future paths. The idea here is that at certain stages in the execution process, when a choice is faced, a branching interpreter, represented by the predicate $\text{branch-exec}()$, is invoked to explore the future for a certain distance and to aid in the decision of what choice to make in the current state. Thus, most of the rules for $\text{branch-exec}()$ are similar to those for $\text{execute}()$; some of the ones that are different are given in figure 3.

$$\begin{aligned} \text{branch-exec}(\varphi \vee \psi, [\varphi]^\cdot L, V_L, b) \\ \Rightarrow \text{branch-exec}(\varphi, L, V_L, b) \wedge \\ \text{branch-exec}(\psi, R, V_R, b) \wedge \\ \text{better}(V_L, V_R) \\ \text{branch-exec}(\varphi \vee \psi, [\psi]^\cdot R, V_R, b) \\ \Rightarrow \text{branch-exec}(\varphi, L, V_L, b) \wedge \\ \text{branch-exec}(\psi, R, V_R, b) \wedge \\ \text{better}(V_R, V_L) \\ \text{branch-exec}(\varphi, [], V, 0) \Leftrightarrow \text{value}(\varphi, V) \\ (\text{branch-exec}(\bigcirc\varphi, L, V, b) \wedge (b > 0)) \\ \Rightarrow \text{branch-exec}(\varphi, L, V, b - 1) \end{aligned}$$

Figure 3: Extra Rules for a Branching Interpreter

The first two rules show that when $\text{branch-exec}()$ executes a choice, both choices are independently executed. A list of choices taken is recorded in the second argument of $\text{branch-exec}()$, and a measure of the “goodness” of the represented path is delivered in the third argument of $\text{branch-exec}()$. The final argument is a bound on the depth of the search.

Thus, during execution, the base interpreter might choose to invoke $\text{branch-exec}()$ at some stage to ex-

⁵Note that this is only possible in a propositional language.

plore the future. Once the future has been simulated for a certain number of states, the values of the **branch-exec()** predicate will represent the possible futures up to that number of states. The base interpreter then chooses which of these possible futures gives the ‘best’ execution and then follows the plan represented by the argument L generated for that particular future. For example, the execution mechanism given earlier using **execute()** might be modified to incorporate the following rules⁶

$$\begin{aligned} \text{execute}(\varphi \vee \psi) \\ \Rightarrow \text{branch-exec}(\varphi \vee \psi, L, V, 8) \wedge \\ \text{planned-exec}(\varphi \vee \psi, L) \end{aligned}$$

Thus, the execution would follow the path represented in L and would be guided down this path by **planned-exec()**. The rules for **planned-exec()** would again be similar to those for **execute()**, the difference being in the rules representing guidance, such as

$$\text{planned-exec}(\varphi \vee \psi, [\varphi]^* L) \Rightarrow \text{planned-exec}(\varphi, L)$$

When the path has been followed as far as possible, execution would revert back to being carried out by **execute()**, i.e.,

$$\text{planned-exec}(\varphi, []) \Rightarrow \text{execute}(\varphi)$$

Thus, heuristics define the choice of which future is ‘best’. For example, it may be that the ‘best’ future is the one with the least number of outstanding constraints.

4.3 A Simple Form of Planning

We can develop a system that incorporates a simple form of planning by utilising the following observations about METATEM.

- In attempting to execute an eventuality, such as $\Diamond p$, the execution mechanism carries out goal-directed forward reasoning (where p represents the goal).
- The meta-level constraints on the execution mechanism effectively prune the search space and direct the execution towards the satisfaction of the eventualities.

Thus, we can implement a naive planning system by

1. representing the basic plan components as facts that are known at the beginning of the execution, and
2. representing schemas for combining separate sub-plans, to produce larger plans, as rules (or meta-rules) in the METATEM program.

⁶Here, the branching interpreter searches forward for at most 8 states.

Given a goal state, characterised by the formula φ , we then try to execute $\Diamond\varphi$. Given the basic execution mechanism, the system will search forward, by constructing linear paths, in an attempt to reach the goal. A more sophisticated system would incorporate the ‘branching look-ahead’ described above. This would generate a branching structure, representing prefixes of all the possible paths that are being explored, and use this structure in deciding which path to follow.

Meta-level constraints can also be added to both prune the search space and direct the execution mechanism towards the goal.

Acknowledgements

This work was supported by ESPRIT under Basic Research Action 3096 (SPEC).

References

- Abadi, M. (1987). *Temporal-Logic Theorem Proving*. PhD thesis, Department of Computer Science, Stanford University.
- Allen, J. F. and Hayes, P. J. (1985). A Common Sense Theory of Time. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 528–531, Los Angeles, California.
- Barringer, H., Fisher, M., Gabbay, D., Gough, G., and Owens, R. (1989). METATEM: A Framework for Programming in Temporal Logic. In *REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness (LNCS Volume 430)*, pages 94–129, Mook, Netherlands. Springer Verlag.
- Caspi, P., Pilaud, D., Halbwachs, N., and Plaice, J. A. (1987). LUSTRE: A Declarative Language for Programming Synchronous Systems. In *Proceedings of the Fourteenth ACM Symposium on the Principles of Programming Languages*, pages 178–188, Munich, West Germany.
- Clarke, E. M. and Emerson, E. A. (1982). Using Branching Time Temporal Logic to Synthesise Synchronisation Skeletons. *Science of Computer Programming*, 2:241–266.
- Fisher, M. (1990). Meta-Programming in METATEM. Temple group report (draft), Department of Computer Science, University of Manchester.
- Fisher, M. D. (1989). Implementing a Prototype METATEM Interpreter. SPEC Project Report, Department of Computer Science, University of Manchester.
- Fujita, M., Kono, S., Tanaka, T., and Moto-oka, T. (1986). Tokio: Logic Programming Language based on Temporal Logic and its compilation into Prolog. In *3rd International Conference on Logic Programming (LNCS Volume 225)*, pages 695–708, London. Springer-Verlag.
- Gabbay, D. (1989). Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems. In Banieqbal, B., Barringer, H., and Pnueli, A., editors, *Proceedings of Colloquium on Temporal Logic in Specification (LNCS Volume 398)*, pages 402–450, Altrincham, U.K. Springer-Verlag.
- Hattori, T., Nakajima, R., Sakuragawa, T., Niide, N., and Takenaka, K. (1986). RACCO: a modal-logic programming language for writing models of real-time process-control systems. Technical report, Research Institute for Mathematical Sciences, Kyoto University.
- Hill, P. M. and Lloyd, J. W. (1988). Analysis of Meta-Programs. In *Proceedings of the Workshop on Meta-Programming in Logic Programming*, pages 27–42, University of Bristol, U.K.
- Moszkowski, B. (1986). *Executing Temporal Logic Programs*. Cambridge University Press, Cambridge, U.K.
- Nilsson, U. and Małuszyński (1990). *Logic, Programming and Prolog*. John Wiley and Sons, Chichester, U.K..
- Owens, R. P. (1990). Using TEQUEL. Internal Report, Department of Computing, Imperial College.
- Perlis, D. (1985). Languages with Self Reference I: Foundations. *Artificial Intelligence*, 25:301–322.
- Pnueli, A. (1977). The Temporal Logic of Programs. In *Proceedings of the Eighteenth Symposium on the Foundations of Computer Science*, Providence.
- Pnueli, A. and Rosner, R. (1989). On the Synthesis of a Reactive Module. In *Proceedings of the 16th ACM Symposium on the Principles of Programming Languages*, pages 179–190.
- Torsun, I. S. and Manning, K. J. (1990). Execution and Application of Temporal Modal Logic. Internal Report, Department of Computing, University of Bradford, U.K.
- van Harmelen, F. (1988). A Classification of Meta-Level Architectures. In *Proceedings of the Workshop on Meta-Programming in Logic Programming*, pages 81–94, University of Bristol, U.K.
- Vardi, M. Y. and Wolper, P. (1986). Automata-theoretic Techniques for Modal Logics of Programs. *Journal of Computer and System Sciences*, 32(2):183–219.

A Definition of MML

A.1 Syntax of MML

The language MML consists of the following symbols

- A set, \mathcal{L}_p , of *predicate symbols*.

Associated with each predicate symbol, p , is a mapping, p_m , from $\{1, \dots, n\}$ (where n is a non-negative integer representing the *arity* of the predicate symbol) to the set of sorts, $\{o, \mu\}$.

Predicates with arity 0 are called propositions.

The set, \mathcal{L}_p , is the disjoint union of the sets \mathcal{L}_{pc} and \mathcal{L}_{pe} , of *component* and *environment* predicate symbols.

- A set, \mathcal{L}_v , of *variable symbols*.

Associated with each variable symbol, v , is an element of the set of sorts, $\{o, \mu\}$, given by $\text{sort}(v)$.

- A set, \mathcal{L}_c , of *constant symbols*⁷.

The *sort* of any element of \mathcal{L}_c is o .

- A set, \mathcal{L}_f , of *function symbols*.

Associated with each function symbol, f , is an arity, given by $\text{arity}(f)$.

- Quantifiers \exists_s, \forall_s , where s is an element of the set of sorts, $\{o, \mu\}$, i.e., $\exists_o, \exists_\mu, \forall_o, \forall_\mu$.

- A set of *propositional connectives*, $\neg, \wedge, \vee, \Rightarrow$.

- A set of *temporal connectives*, $\bullet, \bullet, S, Z, \circlearrowleft, \circlearrowright, U, W, \diamond, \square$.

As there are two disjoint sets of terms, variables are categorised into two sorts, o , representing object-level terms, and μ , representing meta-level terms.

The set of *object terms*, \mathcal{L}_{ot} , contains those terms that are either constants or variables of sort o , or are generated by application of function symbols to other elements of \mathcal{L}_{ot} . Such terms have sort o and can be considered as the basic terms of the language. \mathcal{L}_{ot} is defined as follows.

1. Any constant, c , is in \mathcal{L}_{ot} .
2. Any variable, v , where $\text{sort}(v) = o$, is in \mathcal{L}_{ot} .
3. If o_1, \dots, o_n are in \mathcal{L}_{ot} , and f is a function symbol of arity $n > 0$, then $f(o_1, \dots, o_n)$ is in \mathcal{L}_{ot} .

The set of *meta-level terms*, $\mathcal{L}_{\mu t}$, whose elements have sort μ , contain variables of sort μ , all the formulae constructed from elements of $\mathcal{L}_{\mu t}$, and predicates applied to terms in $\mathcal{L}_{\mu t} \cup \mathcal{L}_{ot}$.

$\mathcal{L}_{\mu t}$ is constructed as follows

1. Any variable, v , where $\text{sort}(v) = \mu$, is in $\mathcal{L}_{\mu t}$.

⁷Only object level constants and functions that apply to object level terms are allowed.

2. If φ and ψ are in $\mathcal{L}_{\mu t}$, then so are

true	false	(φ)	$\neg\varphi$
$\varphi \wedge \psi$	$\varphi \vee \psi$	$\varphi \Rightarrow \psi$	$\varphi \wedge \psi$
$\bullet\varphi$	$\bullet\varphi$	$\varphi S \psi$	$\varphi Z \psi$
$\circlearrowleft\varphi$	$\circlearrowright\varphi$	$\varphi U \psi$	$\varphi W \psi$
$\square\varphi$	$\diamond\varphi$		

3. If x_1, \dots, x_n are in $\mathcal{L}_{ot} \cup \mathcal{L}_{\mu t}$, and p is a predicate symbol of arity $n > 0$, and for each i from 1 to n , $p_m(i) = \text{sort}(x_i)$ then $p(x_1, \dots, x_n)$ is in $\mathcal{L}_{\mu t}$. Note that any proposition symbol is in $\mathcal{L}_{\mu t}$.

Thus, $\mathcal{L}_{\mu t}$ is effectively the set of well-formed formulae (wff) for the object-level, together with meta-level variables.

A term is called *ground* if it contains no variable symbols. Let \mathcal{T} be the subset of $\mathcal{L}_{\mu t}$ containing only ground variables⁸.

The set of well-formed formulae of MML, WFF_m , is defined as follows.

1. If x_1, \dots, x_n are in $\mathcal{L}_{ot} \cup \mathcal{L}_{\mu t}$, and p is a predicate symbol of arity $n > 0$, and for each i from 1 to n , $p_m(i) = \text{sort}(x_i)$ then $p(x_1, \dots, x_n)$ is in WFF_m . Thus, any proposition symbol is in WFF_m .

2. If φ and ψ are in WFF_m , then so are

true	false	(φ)	$\neg\varphi$
$\varphi \wedge \psi$	$\varphi \vee \psi$	$\varphi \Rightarrow \psi$	$\varphi \wedge \psi$
$\bullet\varphi$	$\bullet\varphi$	$\varphi S \psi$	$\varphi Z \psi$
$\circlearrowleft\varphi$	$\circlearrowright\varphi$	$\varphi U \psi$	$\varphi W \psi$
$\square\varphi$	$\diamond\varphi$		

3. If φ is in WFF_m , v is in \mathcal{L}_v , and $\text{sort}(v) = s$, then $\exists_v \varphi$ and $\forall_v \varphi$ are in WFF_m .

A.2 Semantics of MML

Recall that the domain over which terms range is not simply the domain of \mathcal{L}_{ot} — terms also range over the domain of $\mathcal{L}_{\mu t}$ (the names of formulae in MML). Thus, there are two domains: the first is simply the ‘real’ object-level domain, to which elements of \mathcal{L}_{ot} map; the other is the set of names of ground formulae in MML — these are what elements of $\mathcal{L}_{\mu t}$ map to.

The only difficulty here is what to do about variables and predicates. As all variables have a sort, we can bind them to elements of the appropriate domain. The interpretation of a predicate symbol usually gives a signature something like

$$\mathcal{D}^n \rightarrow \{T, F\}.$$

Thus, depending on the values of its arguments, the predicate is interpreted as either T or F . In MML, we

⁸For a more expressive meta-language this restriction that the elements of \mathcal{T} must be grounded can be removed.

also have the possibility of arguments being in $\mathcal{L}_{\mu t}$ (as well as \mathcal{L}_{ot}), so we interpret predicates as

$$(\mathcal{D} \cup \mathcal{T})^n \rightarrow \{T, F\}$$

where \mathcal{T} is the domain associated with $\mathcal{L}_{\mu t}$.

Well-formed formulae of MML are interpreted over model structures of the form

$$\mathcal{M} = \langle S, \mathcal{R}, \mathcal{D}, \mathcal{T}, \pi_c, \pi_f, \pi_p \rangle$$

where

- S is a set of states,
- \mathcal{R} is a relation over S ,
- \mathcal{D} is the object-level domain,
- \mathcal{T} is the meta-level domain (a ground version of $\mathcal{L}_{\mu t}$),
- π_c is a map from \mathcal{L}_c to \mathcal{D} ,
- π_f is a map from \mathcal{L}_f to $\mathcal{D}^n \rightarrow \mathcal{D}$, where n is the arity of f , and,
- π_p is a map from $S \times \mathcal{L}_p$ to $(\mathcal{D} \cup \mathcal{T})^n \rightarrow \{T, F\}$.

A *variable assignment* is a mapping from \mathcal{L}_v to elements of $\mathcal{D} \cup \mathcal{T}$.

Given a variable assignment, V , and the valuation functions, π_c and π_f , associated with a particular model structure, a term assignment τ_{vv} is a mapping from $\mathcal{L}_{ot} \cup \mathcal{L}_{\mu t}$ to $\mathcal{D} \cup \mathcal{T}$ and is defined as follows.

- if $c \in \mathcal{L}_c$, $\tau_{vv}(c) = \pi_c(c)$
- if $f \in \mathcal{L}_f$, and the *arity* of f is n ,
 $\tau_{vv}(f(t_1, \dots, t_n)) = \pi_f(f)(\tau_{vv}(t_1), \dots, \tau_{vv}(t_n))$
- if $v \in \mathcal{L}_v$, $\tau_{vv}(c) = V(c)$
- if $p \in \mathcal{L}_p$, and the *arity* of p is n ,
 $\tau_{vv}(p(t_1, \dots, t_n)) = p(\tau_{vv}(t_1), \dots, \tau_{vv}(t_n))$
- if $t = \text{true}$ or $t = \text{false}$, $\tau_{vv}(t) = t$
- if $t = \text{OP } t'$, where $\text{OP} \in \{\neg, \bigcirc, \odot, \bullet, \odot, \diamond, \square\}$,
 $\tau_{vv}(t) = \text{OP } \tau_{vv}(t')$
- if $t = t' \text{ OP } t''$, where $\text{OP} \in \{\wedge, \vee, \Rightarrow, \mathcal{S}, \mathcal{Z}, \mathcal{U}, \mathcal{W}\}$,
 $\tau_{vv}(t) = \tau_{vv}(t') \text{ OP } \tau_{vv}(t'')$

Note that no checking of the sorts of variables is carried out here; this is taken care of in the semantics of quantification.

The semantics of a well-formed MML formula is given with respect to a model structure, a state at which the formula is to be interpreted, and a variable assignment. The satisfaction relation, \models , relates such tuples to formulae of WFF_m as follows.

- $(\mathcal{M}, s, V) \models \neg \varphi$ iff not $(\mathcal{M}, s, V) \models \varphi$
- $(\mathcal{M}, s, V) \models \varphi \vee \psi$ iff
 $(\mathcal{M}, s, V) \models \varphi$ or $(\mathcal{M}, s, V) \models \psi$
- $(\mathcal{M}, s, V) \models \varphi \wedge \psi$ iff
 $(\mathcal{M}, s, V) \models \varphi$ and $(\mathcal{M}, s, V) \models \psi$
- ETC....**
 $(\mathcal{M}, s, V) \models \forall_\alpha x. \varphi$ iff
for all $d \in \mathcal{D}$,
 $(\mathcal{M}, s, V \upharpoonright [x \mapsto d]) \models \varphi$
- $(\mathcal{M}, s, V) \models \forall_\mu x. \varphi$ iff
for all $t \in \mathcal{T}$,
 $(\mathcal{M}, s, V \upharpoonright [x \mapsto t]) \models \varphi$
- $(\mathcal{M}, s, V) \models \exists_\alpha x. \varphi$ iff
there exists $d \in \mathcal{D}$ such that
 $(\mathcal{M}, s, V \upharpoonright [x \mapsto d]) \models \varphi$
- $(\mathcal{M}, s, V) \models \exists_\mu x. \varphi$ iff
there exists $t \in \mathcal{T}$ such that
 $(\mathcal{M}, s, V \upharpoonright [x \mapsto t]) \models \varphi$
- $(\mathcal{M}, s, V) \models p(x_1, x_2, \dots, x_n)$ iff
 $\pi_p(s, p)(\tau_{vv}(x_1), \tau_{vv}(x_2), \dots, \tau_{vv}(x_n)) = T$

We will usually deal with closed formulae, i.e., formulae containing no free variables. In this case, we use the empty mapping, $[]$, as the initial variable assignment.

In summary, the main difference between MML and standard temporal logics is that the interpretation of predicate symbols is not just a map from element of the base domain to $\{T, F\}$, but is a map from elements of either the base domain or the domain of names of formulae (\mathcal{T}) to $\{T, F\}$. (This is why, in the example that follows, π_p maps formulae, as well as terms, to $\{T, F\}$.)

A.3 Example

We now give an example of a model for an MML formula and show how the semantics can be used to show that the formula is satisfied in the model. This example is relevant to the applications described in section 4.

Consider the model

$$\begin{aligned} D &= \{\}, \\ T &= \{r, q, \bigcirc r, \text{etc}\}, \\ \pi_c &= [], \\ \pi_f &= [], \\ \pi_p &= [0 \mapsto \{\text{execute}(\bigcirc r), \text{execute}(\bigcirc q)\}] \\ &\quad [1 \mapsto \{\text{execute}(r), \text{execute}(\bigcirc q), \text{execute}(q)\}] \\ &\quad \dots \text{etc....} \end{aligned}$$

Here, we use an abbreviated representation of π_p , where each state maps to the predicate applications that are true in that state. We now consider the satisfaction of the following formula in the above model when interpreted at state 1.

$$\forall_{\mu}x. \bullet\text{execute}(\bigcirc x) \Rightarrow \text{execute}(x)$$

The model satisfies the formula if

$$(\mathcal{M}, 1, []) \models \forall_{\mu}x. \bullet\text{execute}(\bigcirc x) \Rightarrow \text{execute}(x).$$

This is satisfied if for all t in T ,

$$(\mathcal{M}, 1, [x \mapsto t]) \models \bullet\text{execute}(\bigcirc x) \Rightarrow \text{execute}(x).$$

So, for all t in T ,
if

$$(\mathcal{M}, 1, [x \mapsto t]) \models \bullet\text{execute}(\bigcirc x)$$

then

$$(\mathcal{M}, 1, [x \mapsto t]) \models \text{execute}(x).$$

Assuming that $0 \neq 1$, this reduces to,
if

$$(\mathcal{M}, 0, [x \mapsto t]) \models \text{execute}(\bigcirc x)$$

then

$$(\mathcal{M}, 1, [x \mapsto t]) \models \text{execute}(x),$$

and finally to,
if

$$\pi_p(0, \text{execute})(\bigcirc t) = T$$

then

$$\pi_p(1, \text{execute})(t) = T.$$

Now, in state 0, $\text{execute}(\bigcirc t)$ is true for $t \in \{r, q\}$. Thus we must show that in state 1, both $\text{execute}(r)$ and $\text{execute}(q)$ are true. As this is so, the model satisfies the formula (at least when interpreted at state 1).

Further examples and details of both MML and FML* are given in (Fisher, 1990).

Pragmatic Logics

John Bell
 Computer Science Department
 Queen Mary and Westfield College
 University of London
 London E1 4NS
 England

Abstract

This paper defines and investigates pragmatic logics. These are a more general than Shoham's preference logics and the cumulative logics of Kraus, Lehmann and Magidor. It begins with the semantic definition of pragmatic logics, and logics such as Autoepistemic Logic and Default Logic are shown to be pragmatic logics. A proof theory for pragmatic logics is developed and a general completeness theorem is proved for them. The proof can be used to show the completeness of particular pragmatic logics, and this is done in the case of Autoepistemic Logic. The logical properties of pragmatic logics are then investigated, and it is shown that a hierarchy of pragmatic logics can be defined on this basis. Next, the close relations between pragmatic logics and conditional logics are discussed. Many well known pragmatic logics are then located in the hierarchy developed earlier, and the paper concludes with some general remarks on the nature of pragmatic logics.

1. Pragmatic Reasoning

Deductive reasoning is *analytic* : the conclusions which follow by deductive reasoning from a set of premises are "contained in" the premises (Kant, 1781). Deductive reasoning is therefore sound. However there is a sense in which it is uninformative : no amount of deductive reasoning can produce new information. Of course it is necessary here to distinguish between ideal and non-ideal reasoners. For example, Chess can be viewed as a finite, and so decidable, deductive system. Consequently the outcome of every Chess position is known to ideal deductive reasoners who know the rules of Chess (Konolige, 1984), and as a result Chess is of little interest to them. However the outcome of most Chess positions is unknown to non-ideal reasoners and may as a result be of great interest to them.

Deductive reasoning is useful in everyday life. However in everyday life agents need to infer things that even ideal deductive agents cannot infer. For example, in order to form even a simple plan to catch a bus the agent needs to infer that it is safe to do so (given that it is usually safe to do so), that the bus will actually come at something like the advertised time, etc. Reasoning of this kind is *synthetic*; that is, it "goes beyond" the premises (Kant, 1781). Synthetic reasoning includes inductive and probabilistic reasoning as well as commonsense reasoning. When judged by the standards of deductive reasoning, synthetic reasoning is unsound. From a practical viewpoint it is considered as defeasible, useful and indeed often essential. This paper is concerned with a qualitative species of synthetic reasoning which will be called *pragmatic reasoning*. Unlike deductive reasoning, pragmatic reasoning is particular to a context; it is therefore pragmatic in the sense of (Montague 1974) and (Allen, 1987). Pragmatic reasoning is also practical. It typically occurs in a context in which information is limited and employs heuristic rules for arriving at reasonable conclusions given the context and its limitations; so it also pragmatic in the sense of (Levinson, 1983). These rules are, in effect, meta-rules for reasoning about particular contexts. For example our agent might reason that it is safe to catch the bus as it is usually safe to do so and there is no reason to think otherwise; that is, there is nothing about the particular context which leads the agent to think otherwise. Similarly, the agent might infer that the bus will be on time as the bus normally comes on time and there is no reason, in the particular context, to suppose that it won't. If the context changes, say the agent learns of a bus strike, these conclusions have to be revised. Pragmatic reasoning is thus context-sensitive and the principle of Monotony:

If A implies C then A and B imply C

which holds for context-free deductive reasoning fails for pragmatic reasoning.

There are many well known formalisations of aspects of pragmatic reasoning; better known as nonmonotonic logics. Among the best known are Predicate Circumscription (McCarthy, 1980), Default Logic (DL) (Reiter, 1980) and Autoepistemic Logic (AEL) (Moore, 1983). These all involve reasoning in a context as well as meta-level reasoning about it. The circumscription of a predicate P in the context of an axiom A allows us to conclude that the individuals satisfying P in A are the only individuals which satisfy P . In the case of a default theory (D, W) , the context consists of the world description W and the default rules D which sanction inferences in it. Similarly in AEL the context is an autoepistemic theory A and an autoepistemic expansion of A is a superset of reasonable beliefs given this context.

Generalising an idea of McCarthy (McCarthy, 1980) and (Davis, 1980), Shoham (Shoham, 1987) produced a general semantic framework for a subclass of pragmatic logics which he called *preference logics*. The idea is to take an existing logic L and to define a strict partial order $<$, called a preference criterion, on the class of all its models. The preferred models in which A is true (A -models) are then defined as to be the highest A -models in the ordering, and A is said to preferentially entail B iff the preferred A -models are all B -models. The resulting logic is said to be a preference logic. This framework is general enough to incorporate Predicate Circumscription and several other nonmonotonic logics. Unfortunately however it cannot incorporate logics like AEL and DL as these are not cumulative (Makinson, 1989). However it is possible to generalise Shoham's idea by replacing the preference criterion with a function which selects the A -models considered relevant on pragmatic grounds, and A is said to pragmatically entail B iff the relevant A -models are all B -models. The resulting logic is said to be a pragmatic logic.

A formal definition of pragmatic logics is given in the next section and logics such as AEL and DL are shown to be pragmatic logics. Section 3 develops a proof theory for pragmatic logics and proves a general completeness theorem for them. The proof can be used to show the completeness of particular pragmatic logics, and this is done in the case of AEL. In Section 4 the logical properties of pragmatic logics are investigated, and it is shown that a hierarchy of pragmatic logics can be defined on this basis. Pragmatic logics include the cumulative logics studied by (Kraus, Lehmann and Magidor, 1990), so the work here complements theirs. Many of the ideas in the development of preference logics, the most obvious being the use of a selection function, come from conditional logic, and particularly from the work of

Lewis (Lewis, 1973). The connections between pragmatic logics and conditional logics are discussed in Section 5. Many well known pragmatic logics are then located in the hierarchy developed in section 4, and the paper concludes with some general remarks on the nature of pragmatic logics.

2. Model theory

Let L be a logic with language L and consequence relation \models_L ; L need not be closed under Boolean connectives and \models_L need not be classical (or even monotonic). It is assumed that L contains the sentential constants *true* and *false* denoting "true" and "false" respectively. Let I be the non-empty set of all models of L (it is assumed that L is consistent), and let $[A]^L$ denote the set of all A -models of L ; that is, the set $\{i \in I \mid i \models_L A\}$ of models of L in which A is true. Then we can define a new logic L_M called a *pragmatic logic* as follows:

Definition 2.1. A *pragmatic model* is a triple:

$$M = (I, []^M, []^M) \text{ where:}$$

I is the set of all models of L ,

$[]^M: L \rightarrow \mathcal{P}(I)$ is such that $[A]^M = [A]^L$, and

$[]^M: L \rightarrow \mathcal{P}(I)$ is such that $[A]^M \subseteq [A]^L$.

$[]^M$ is said to be a (*semantically based*) *selection function*. For a given sentence A , $[]^M$ selects the *relevant* A -models in M ; where relevance is decided on pragmatic grounds.

The standard Tarskian notions of satisfaction, entailment and validity for L_M can be defined using $[]^M$.

Definition 2.2. For any $A, B \in L$:

$$\begin{array}{lll} M, i \models_{L_M} A & \text{iff} & i \in [A]^M \\ A \models_{L_M} B & \text{iff} & [A]^M \subseteq [B]^M \\ \models_{L_M} A & \text{iff} & I \subseteq [A]^M \end{array}$$

If $M, i \models_{L_M} A$ we say that A is satisfied at i in M , if $A \models_{L_M} B$ we say that A entails B in L_M and if $\models_{L_M} A$ we say that A is valid in L_M . As $\models_{L_M} A$ iff *true* $\models_{L_M} A$ the validity relation of L_M is a subset of the entailment relation, and we can regard $\models_{L_M} A$ as an abbreviation for *true* $\models_{L_M} A$. Clearly if $[A]^M$ is empty then $[A]^M = [\text{false}]^M$ and $A \models_{L_M} B$ for any B .

It is often interesting to distinguish a set of pragmatically minimal models in M .

Definition 2.3. Let $I_{\perp} = [\text{true}]^M$.

The new notions of satisfaction, entailment and validity for L_M can now be defined in terms of $[]^M$.

Definition 2.4. For any $A, B \in L$:

$$\begin{array}{lll} M, i \models_{L_M} A & \text{iff} & i \in [A]^M \\ A \models_{L_M} B & \text{iff} & [A]^M \subseteq [B]^M \\ \models_{L_M} A & \text{iff} & I_{\perp} \subseteq [A]^M \end{array}$$

If $M, i \models_{L_M} A$ we say that i *pragmatically satisfies* A in M , if $A \models_{L_M} B$ we say that A *pragmatically entails* B in L_M , and if $\models_{L_M} A$ we say that A is *pragmatically valid* in L_M . As a consequence of these definitions $\models_{L_M} \subseteq \models_{L_M}$ in each case. As $\models_{L_M} A$ iff $\text{true} \models_{L_M} A$ the pragmatic validity relation of L_M is a subset of its pragmatic entailment relation. Clearly if $[A]^M$ is empty then $[A]^M = [\text{false}]^M$ and $A \models_{L_M} B$ for any B . If $[]^M = []^M$ then the notions introduced in the two definitions co-incide. So any logic with a suitable model theory is a pragmatic logic. The interesting pragmatic logics are those where $[]^M$ and $[]^M$ differ.

We have defined a *skeptical* form of pragmatic entailment. At least two alternatives are suggested by the literature. The *credulous* form requires that some selected A -model is a B -model:

$$A \models_{oL_M} B \quad \text{iff} \quad [A]^M \circ [B]^M$$

where $S \circ T$ is true iff the sets S and T overlap; that is, iff their intersection is non-empty. If L contains classical negation, the skeptical and credulous forms are inter-definable as:

$$\begin{array}{lll} A \models_{oL_M} B & \text{iff} & A \not\models_{L_M} \neg B \\ A \models_{L_M} B & \text{iff} & A \not\models_{oL_M} \neg B \end{array}$$

So either can be taken as primitive. The *choice* form of pragmatic entailment requires that a chosen relevant A -model is also a B -model; that is:

$$A \models_{eL_M} B \quad \text{iff} \quad f([A]^M) \subseteq [B]^M$$

where $f(S)$ is a singleton containing the chosen member of S , if S is non-empty, and the empty set otherwise. Choice entailment can thus be regarded as the special case of skeptical entailment where $[A]^M$ contains at most one model. The three notions are related as follows:

$$A \models_{oL_M} B \quad \text{if} \quad A \models_{eL_M} B$$

And if $[A]^M$ is non-empty then:

$$A \models_{oL_M} B \quad \text{if} \quad A \models_{eL_M} B$$

In this paper the skeptical form is taken as the central notion of pragmatic entailment. It is more general than choice entailment. It is also consistent in the non-vacuous case, whereas credulous entailment need not be.

Definition 2.4. If M is a pragmatic model and L_M is the logic defined in terms of M , then L_M is a *pragmatic logic* and M is the *defining model* of L_M .

Examples 2.5.

1. The preference logic $L_<$ (Shoham, 1987) is defined by taking a logic L over a language L and defining the strict partial order $<$ on the non-empty set I of all L -models. We can therefore define the corresponding pragmatic logic L_M by defining $[A]^M$, for each $A \in L$, to be the set of preferred A -models. For example, the Logic of Minimal Knowledge (LMK) of (Halpern and Moses, 1984) can be characterised as a pragmatic logic. L is the modal logic S5, $[A]^M$ is the set of preferred A -models of S5 as defined in (Shoham, 1987); I_{\perp} contains only the canonical model of S5. For the circumscription of a predicate P (McCarthy, 1980), L is first-order predicate calculus (FOPC) and I is the class of all models of FOPC. $[A]^M$ is the set of preferred A -models of FOPC as defined in (Shoham, 1987); I_{\perp} then contains all FOPC-models in which the extension of P is empty.

2. More generally, the cumulative logics of (Kraus, Lehmann and Magidor, 1990) are pragmatic logics. A cumulative model is a triple $(S, I, <)$ where S is a set of states, I is a function which labels each state with an S5 model, and $<$ is a smooth asymmetric binary relation on S . We can therefore define the corresponding pragmatic model by letting I be the set of states, $[A]^M$ be the set \hat{A} of states satisfying A , and $[A]^M$ be the set of $<$ -minimal states in \hat{A} .

3. AEL can also be incorporated. (Moore, 1984) shows that every stable expansion (Konolige: 'AE extension') of an autoepistemic theory has what he terms a possible-worlds model. Each possible-worlds model has a set of corresponding KD45 models (Bell, 1990a). So L is KD45, I is the set of all KD45-models, and $[A]^M$ is the set of KD45 models corresponding to the possible-worlds models of A ; I_{\perp} contains only the canonical model of KD45. As with DL, the possibility of multiple extensions leads to a choice in the definition AE inference. Specifically there are skeptical, choice and credulous forms of AE inference corresponding to the three forms of pragmatic entailment discussed.

4. By virtue of the result of (Konolige, 1987) the case for DL is the same as that for AEL except that $[A]^M$ is now defined to be the set of KD45-models corresponding to the AE extensions of A that are strongly grounded in A.

5. (Brown and Shoham, 1989) show how to define modal models for theories of JTMS (Doyle, 1979). They leave open the choice of the underlying modal logic. So let L be the chosen modal logic, and let $[A]^M$ be the set of L-models of A as defined in (Brown and Shoham, 1989); I_\perp contains only the canonical model of L.

The examples suggest definite senses for ‘minimal’ and ‘relevant’. The minimal models specify (are models of) a base logic and the relevant A-models are those A-models which otherwise differ minimally, in some sense, from the minimal models. In the case of the circumscription of a predicate P the minimal models are those in which the extension of P is empty, so the base logic is FOPC together with the circumscription of P in FOPC, and the relevant A-models are those A-models in which the extension of P is otherwise minimal. For LMK the minimal model i is the canonical model of S5 representing the state of maximal ignorance in which only the theorems of S5 are known and the relevant A-models are those which otherwise differ minimally from i in terms of what is known. In the case of AEL the minimal model i is the canonical model of KD45 representing the maximally skeptical belief state in which nothing but the theorems of KD45 are believed so the base logic is KD45 and the relevant A-models are among those in which A is believed and which otherwise differ minimally from i in terms of what is believed.

The examples also show that pragmatic logics include cumulative logics; and hence preference logics. Pragmatic logics also include logics like AEL and DL which are not cumulative (Makinson, 1989). So the inclusion of cumulative logics is proper.

3. Proof theory

Definition 3.1. Let $g: I \rightarrow \wp(L)$ be such that:

$$g(i) = \{A \mid i \in [A]^M\}.$$

Definition 3.2. Let $M = (I, []^M, []^M)$ be the defining model of a pragmatic logic. Then:

$$I' = \{g(i) \mid i \in I\}$$

The members of I' are called *extensions*.

There is seldom an isomorphism between I and I' as it is usually the case that many models can give rise to the same extension; although of course the converse cannot be the case.

Definition 3.3. A function $[]^{M'}$ of type $L \rightarrow \wp(I')$ is called a (*syntactic based*) *pragmatic selection function*.

Definition 3.4. The syntactically based selection function $[]^{M'}$ corresponds to the semantically based selection function $[]^M$ iff for any $A \in L$:

$$[A]^{M'} = \{g(j) \mid j \in [A]^M\}$$

Definition 3.5. Let L_M be a pragmatic logic with defining model $M = (I, []^M, []^M)$. Then the *canonical model* $M' = (I', []^{M'}, []^{M'})$ of L_M is defined as follows:

$$I' = \{g(i) \mid i \in I\},$$

$$[A]^{M'} = \{i' \in I' \mid A \in i'\} \text{ for all } A \in L, \text{ and}$$

$[]^{M'}$ is the syntactic selection function corresponding to $[]^M$.

Proposition 3.6. Every pragmatic logic L_M with defining model $M = (I, []^M, []^M)$ has a unique canonical model $M' = (I', []^{M'}, []^{M'})$.

Proof. $[]^{M'}$ and $[]^M$ are both well-defined and unique. \square

Definition 3.7. The provability relation \vdash_{L_M} of L_M is defined in terms of its canonical model M' as follows:

$$A \vdash_{L_M} B \quad \text{iff} \quad [A]^{M'} \subseteq [B]^{M'}$$

If $A \vdash_{L_M} B$ we say that B is *pragmatically inferrable* from A in L_M . Credulous and Choice forms of pragmatic inference can, of course, also be defined; as can the set I_\perp of minimal extensions. Note that $\vdash_L \subseteq \vdash_{L_M}$.

Lemma 3.8. Let L_M be a pragmatic logic with defining model M and canonical model M' . Then for any $A, B \in L$:

$$[A]^M \subseteq [B]^M \quad \text{iff} \quad [A]^{M'} \subseteq [B]^{M'}$$

Proof.

$$\begin{aligned} [A]^M \subseteq [B]^M &\quad \text{iff} \quad \{j \mid j \in [A]^M\} \subseteq [B]^M \\ &\quad \text{iff} \quad \{g(j) \mid j \in [A]^M\} \subseteq [B]^{M'} \\ &\quad \text{iff} \quad [A]^{M'} \subseteq [B]^{M'} \end{aligned} \quad \square$$

Theorem 3.9. (Soundness and Completeness) For any pragmatic logic L_M and any $A, B \in L$:

$$A \Vdash_{L_M} B \text{ iff } A \vdash_{L_M} B$$

Proof.

$$\begin{aligned} A \Vdash_{L_M} B &\text{ iff } [A]^M \subseteq [B]^M \\ &\text{ iff } [A]^{M'} \subseteq [B]^{M'} \text{ (Lemma 3.8)} \\ &\text{ iff } A \vdash_{L_M} B \quad \square \end{aligned}$$

So in order to show the completeness of a particular pragmatic logic it is sufficient to provide the semantic and syntactic components and to ensure that the selection functions correspond.

Example 3.10. Here are the details for AEL. The function g maps KD45-models in I to the corresponding sets of sentences. For each $A \in L$, $[A]^{M'}$ is the set of AE extensions of A . The function $h : I' \rightarrow \wp(I)$ maps AE extensions in I' to the corresponding sets of KD45-models as indicated in Example 2.5.3. $[]^M$ can now be formally defined, for each $A \in L$, as follows: $[A]^M = \{i \in h(i') \mid i' \in [A]^{M'}\}$. As $i' \in [A]^{M'} \iff \exists i. i \in h(i') \wedge i \in [A]^M \iff \exists i. g(i) = i' \wedge i \in [A]^M \iff i' \in \{g(i) \mid i \in [A]^M\}$, $[]^{M'}$ corresponds to $[]^M$. So completeness follows from Theorem 3.9.

4. Logical properties

This section considers the logical properties of pragmatic logics. Throughout, it is assumed for simplicity that L is a (consistent) logic (with language L) which contains PC and is compact. The symbols \vdash and \Vdash are used to denote the consequence relation of PC. The proof techniques used are based on those of (Lewis, 1973) and (Kraus, Lehmann and Magidor, 1990).

Definition 4.1. A logic L is a B -logic iff its consequence relation satisfies the following properties:

$$A \Vdash A \quad (\text{Reflexivity})$$

$$A \Vdash B, \vdash B \supset C / A \Vdash C \quad (\text{Right Weakening})$$

$$A \Vdash B, A \Vdash C / A \Vdash B \wedge C \quad (\text{And})$$

Lemma 4.2. The following is a derived rule of B -logics:

$$\vdash B_1 \wedge \dots \wedge B_n \supset C, A \Vdash B_1, \dots, A \Vdash B_n / A \Vdash C \quad (\text{for } n \geq 0) \quad (\text{General Right Weakening})$$

Proof. For $n = 0$ suppose that $\vdash C$. Then $\vdash A \supset C$, so, by Reflexivity and Right Weakening, $A \Vdash C$. For $n \geq 1$ suppose that $\vdash B_1 \wedge \dots \wedge B_n \supset C$ and that $A \Vdash B_1, \dots, A \Vdash B_n$. By $n-1$ applications of And we have $A \Vdash B_1 \wedge \dots \wedge B_n$. So, by Right Weakening, we have $A \Vdash C$. \square

Lemma 4.3. (Soundness) If \Vdash_{L_M} is the consequence relation of a pragmatic logic then \vdash_{L_M} is the consequence relation of a B -logic; that is, the rules of B are satisfied by the consequence relations defined by pragmatic models.

Proof. The proof is routine. We illustrate with the case for Right Weakening. Suppose that $[A]^M \subseteq [B]^M$ and $\vdash B \supset C$. From the second assumption and the soundness of PC we have we have $\vdash B \supset C$, and hence $[B]^M \subseteq [C]^M$. So it follows from the first assumption that $[A]^M \subseteq [C]^M$. \square

Turning to completeness, let L be a B -logic with consequence relation \Vdash_L . We will define a pragmatic logic with consequence relation \Vdash_{L_M} and show that $\Vdash_L = \Vdash_{L_M}$.

Let I be the set of all models of L , let $[A]^L$ denote the set of L -models in which A is true, and let $\Theta(A) = \{B \mid A \Vdash_L B\}$ be the set of \Vdash_L -consequences of A .

Definition 4.4. A model $i \in I$ is *normal for* A iff $\Theta(A) \subseteq g(i)$. The set of normal L -models of A is denoted $[A]^L$.

Lemma 4.5. Suppose that L is a B -logic with consequence relation \Vdash_L . Then, for any $A, B \in L$:

$$[A]^L \subseteq [B]^L \text{ iff } A \Vdash_L B$$

Proof. The if part follows from Definition 4.4. For the only if part suppose that $A \Vdash_L B$. Then there is a normal model i for A that does not satisfy B . In particular $\Gamma_0 = \{\neg B\} \cup \Theta(A) \subseteq g(i)$. To show that i exists it is sufficient to show that Γ_0 is L -satisfiable. Suppose not. Then, by the compactness of L , there exists a finite subset of Γ_0 that is not L -satisfiable, and therefore a finite set $\{D_1, \dots, D_n\} \subseteq \Theta(A)$ such that $\vdash D_1 \wedge \dots \wedge D_n \supset B$. By definition of Θ we have $A \Vdash_L D_i$. So by General Right Weakening we have $A \Vdash_L B$ contradicting the original assumption. \square

The model $M = (I, []^M, []^M)$ is now defined as follows. Let:

$$I = \{i \mid i \text{ is an } L\text{-model}\},$$

$$[A]^M = [A]^L, \text{ and}$$

$$[A]^M = [A]^L.$$

Lemma 4.6. M is the defining model of a Pragmatic Logic.

Proof. As L is a B -logic, \vdash_L satisfies Reflexivity. So, for any $A \in I$, $[A]^L \subseteq [A]^L$. \square

Lemma 4.7. Let \vdash_{LM} be the consequence relation defined by M . Then:

$$A \vdash_{LM} B \text{ iff } A \vdash_L B.$$

Proof.

$$A \vdash_{LM} B \text{ iff } [A]^M \subseteq [B]^M$$

$$\text{iff } [A]^L \subseteq [B]^L$$

$$\text{iff } A \vdash_L B \quad (\text{Lemma 4.5}) \quad \square$$

Theorem 4.8. (Basic Representation Theorem) \vdash_{LM} is the consequence relation of a B -logic iff \vdash_{LM} is the consequence relation of a pragmatic logic.

Proof. The if part is soundness. The only if part follows from Lemma 4.6, and Lemma 4.7. \square

The following semantic conditions correspond to the rules of B -logics:

$$[A] \subseteq [A] \quad (\text{reflexivity})$$

$$\text{If } [A] \subseteq [B] \text{ and } \vdash B \supset C \text{ then } [A] \subseteq [C] \quad (\text{right weakening})$$

$$\text{If } [A] \subseteq [B] \text{ and } [A] \subseteq [C] \text{ then } [A] \subseteq [B \wedge C] \quad (\text{and})$$

Many other conditions can be imposed. The following are of particular interest:

Definition 4.9. Conditions on pragmatic models:

$$\text{If } \vdash A = B \text{ then } [A] = [B] \quad (\text{left logical equivalence})$$

$$\text{If } I_\perp \subseteq [A] \text{ then } [A] \subseteq I_\perp \quad (\text{conjunctive sufficiency})$$

$$I_\perp \cap [A] \subseteq [A] \quad (\text{weak conditionalization})$$

$$\text{If } [A] \subseteq [B] \text{ and } [A \wedge B] \subseteq [C] \text{ then } [A] \subseteq [C] \quad (\text{cut})$$

$$\text{If } [A] \subseteq [B] \text{ and } [A] \subseteq [C] \text{ then } [A \wedge B] \subseteq [C] \quad (\text{cautious monotony})$$

$$\text{If } [A] \subseteq [C] \text{ and } [B] \subseteq [C] \text{ then } [A \vee B] \subseteq [C] \quad (\text{or})$$

$$\text{If } [A] \not\subseteq [B] \text{ and } [A] \subseteq [C] \text{ then } [A \wedge B] \subseteq [C] \quad (\text{rational monotony})$$

$$\text{If } [A] \not\subseteq [B] \text{ then } [A] \subseteq [B] \quad (\text{excluded middle})$$

$$\text{If } [A] \subseteq [C] \text{ then } [A \wedge B] \subseteq [C] \quad (\text{monotony})$$

The conditions conjunctive sufficiency and weak conditionalization impose restrictions on the set of minimal models: together they imply that $[A] = I_\perp$ if $I_\perp \subseteq [A]$. Note that rational monotony and excluded middle differ from the other conditions in that they involve both skeptical and credulous inference, as can be seen from the following reformulations using the overlap operator:

$$\text{If } [A] \circ [B] \text{ and } [A] \subseteq [C] \text{ then } [A \wedge B] \subseteq [C] \quad (\text{rational monotony})$$

$$\text{If } [A] \circ [B] \text{ then } [A] \subseteq [B] \quad (\text{excluded middle})$$

Definition 4.10. Corresponding inference rules:

$$I \dashv A = B, A \vdash C / B \vdash C \quad (\text{Left Logical Equivalence})$$

$$I \dashv A \wedge B / A \vdash B \quad (\text{Conjunctive Sufficiency})$$

$$A \vdash B / I \dashv A \supset B \quad (\text{Weak Conditionalization})$$

$$A \vdash B, A \wedge B \vdash C / A \vdash C \quad (\text{Cut})$$

$$A \vdash B, A \vdash C / A \wedge B \vdash C \quad (\text{Cautious Monotony})$$

$$A \vdash C, B \vdash C / A \vee B \vdash C \quad (\text{Or})$$

$$A \vdash \neg B, A \vdash C / A \wedge B \vdash C \quad (\text{Rational Monotony})$$

$$A \vdash \neg B / A \vdash B \quad (\text{Excluded Middle})$$

$$A \vdash C / A \wedge B \vdash C \quad (\text{Monotony})$$

Rational Monotony and Excluded Middle can be reformulated as follows:

$$A \vdash_o B, A \vdash C / A \wedge B \vdash C \quad (\text{Rational Monotony})$$

$$A \vdash_o B / A \vdash B \quad (\text{Excluded Middle})$$

Lemma 4.11. (Extended Soundness) Let C be a set of conditions from Definition 4.9 and let R be the corresponding set of rules from Definition 4.10. If \vdash_{LM} is the consequence relation of a pragmatic logic whose defining model satisfies C then \vdash_{LM} is the consequence relation of a $B+R$ -logic; that is, the rules $B+R$ are

satisfied by the relations defined by pragmatic models satisfying C.

Proof. It is sufficient to show that the lemma holds for each rule in R given the corresponding condition in C. We illustrate with a single case. For Weak Conditionalization suppose that \vdash_{L_M} is the consequence relation defined by a pragmatic model M, and that M satisfies weak conditionalization. If $[A]^M \subseteq [B]^M$ it follows by weak conditionalization that $I_\perp \cap [A]^M \subseteq [B]^M$, so $I_\perp \subseteq [A \supset B]^M$. \square

To show completeness, suppose that L is a logic and that its consequence relation \vdash_L satisfies some of the rules stated in Definition 4.10. We need to show that its defining model M satisfies the corresponding conditions from Definition 4.9.

Lemma 4.12. Let L be a B+R logic; where R is a set of rules from Definition 4.10. Then M satisfies the corresponding set C of conditions from Definition 4.9.

Proof. It is sufficient to show that the lemma holds for each condition C given the corresponding rule R. We give two cases only.

For left logical equivalence suppose that $\vdash A \equiv B$. Consider any $C \in L$ such that $[A]^M \subseteq [C]^M$. Then, by completeness of PC, $\vdash_A \equiv B$ and, by definition of M and Lemma 4.5, $A \vdash_L C$. By Left Logical Equivalence then we have $B \vdash_L C$. So it follows from the definition of Θ that $\Theta(A) \subseteq \Theta(B)$. By a similar argument we can show that $\Theta(B) \subseteq \Theta(A)$. So $[A]^L = [B]^L$, and the result follows by definition of M.

For conjunctive sufficiency suppose that $I_\perp \subseteq [A]^M$. Consider any $B \in \Theta(\text{true})$. By definition of I_\perp , $I_\perp \subseteq [B]^M$. With the original supposition this gives $I_\perp \subseteq [A \wedge B]^M$. By Lemma 4.5 we then have $\text{true} \vdash_L A \wedge B$ and so $\vdash_L A \wedge B$. By Conjunctive Sufficiency, we have $A \vdash_L B$. So it follows from the definition of Θ that $B \in \Theta(A)$. So $\Theta(\text{true}) \subseteq \Theta(A)$. So $[A]^M \subseteq [\text{true}]^M$ and the conclusion follows by definition of M. \square

Theorem 4.13. (Extended Representation Theorem) Let C be a set of conditions from Definition 4.9 and let R be the corresponding set of rules from Definition 4.10. Then \vdash_{L_M} is the consequence relation of a B+R-logic iff \vdash_{L_M} is the consequence relation defined by a pragmatic model M which satisfies the conditions in C.

Proof. The if part is Lemma 4.11. The only if part follows from Theorem 4.8, and lemmas 4.6 and 4.12. \square

B is the weakest pragmatic logic based on PC. It is possible to define a hierarchy of increasingly strong logics.

Definition 4.14. Hierarchy of pragmatic logics containing PC:

$$B = B + \text{Left Logical Equivalence},$$

$$C = C + \text{Cut} + \text{Cautious Monotony},$$

$$P = C + \text{Or},$$

$$R = P + \text{Rational Monotony},$$

$$E = R + \text{Excluded Middle},$$

$$M = E + \text{Monotony}.$$

There is some redundancy in these definitions; for example, And is derivable in C, Cut in P, and Rational Monotony in E. As a result, the systems C, P and R thus correspond to their namesakes in (Lehmann, 1989) and (Kraus, Lehmann and Magidor, 1990). Conjunctive Sufficiency is derivable in C, and Weak Conditionalization is derivable in P.

5. Relation to conditional logic

Much of this will seem familiar to readers with a knowledge of conditional logic as pioneered by Stalnaker (Stalnaker, 1968) and Lewis (Lewis, 1973), and surveyed by Nute (Nute, 1984). Stalnaker distinguished the *logical* problem of conditionals, that of giving the truth conditions for conditionals and logics of them, from the *pragmatic* problem of conditionals, the problem of specifying how to evaluate particular conditionals.

On the logical side an intensional conditional operator, \Rightarrow say, is added to a classical language formed in the usual way from a set P of primitive propositions:

$$\text{If } A \in L \text{ and } B \in L \text{ then } A \Rightarrow B \in L$$

Note that the conditional operator can be iterated.

Selection functions are often used to give semantics for conditionals. For the sake of definiteness, a model is a triple $M = (I, V, f)$ where:

I is a non-empty set of possible worlds,

V: $P \times I \rightarrow \{\text{true, false}\}$ is a valuation, and

f: $L \times I \rightarrow \wp(I)$ is a selection function.

The truth condition is then given, for a model M and world i in M , as follows:

$$M, i \models A \Rightarrow B \text{ iff } f(A, i) \subseteq [B]^M$$

Intuitively, f selects the closest A -worlds to i in M . Conditionals of this kind are not monotonic; they do not have the property:

$$(A \Rightarrow C) \supset (A \wedge B \Rightarrow C)$$

Further non-properties include transitivity and contraposition. The positive logical properties of conditionals of this kind result, as in pragmatic logics, from conditions imposed on the selection function. For example, cut of Definition 4.9 becomes:

If $f(A, i) \subseteq [B]$ and $f(A \wedge B, i) \subseteq [C]$

$$\text{then } f(A, i) \subseteq [C]$$

with corresponding axiom:

$$(A \Rightarrow B) \wedge (A \wedge B \Rightarrow C) \supset (A \Rightarrow C)$$

Comparing this axiom with the rule Cut makes the logical connection between conditional and pragmatic logics clear :

$$A \Vdash B, A \wedge B \Vdash C / A \Vdash C$$

The object-level conditional operator \Rightarrow encodes the meta-level pragmatic entailment symbol \Vdash , so meta-level properties of pragmatic logics can be stated in the object-language of conditional logics. This correspondence was first noted by Van Benthem (Shoham, 1987) and developed in the context of predicate circumscription in (Van Benthem, 1988). It was also developed independently in the context of preference logics in (Bell, 1990a) and generalised further in (Bell, 1990c).

Complications arise because the conditional operator can be iterated. Furthermore, some axioms of conditional logic have no pragmatic logic counterparts. For example, the "centering" axioms of conditional logic:

$$(A \wedge B) \supset (A \Rightarrow B) \quad (\text{CS})$$

$$(A \Rightarrow B) \supset (A \supset B) \quad (\text{MP})$$

are similar to, but not exact translations of, Conjunctive Sufficiency and Weak Conditionalization.

Definition 5.1. (Makinson, 1990) Let A_i and B_i be sentences constructed from atomic sentences using only truth-functional connectives, and let σ_i be either null or negation. Let \Vdash be the consequence relation of a pragmatic logic L_M and let L_C be a conditional logic. Then there is a Boolean correspondence between L_M and L_C if:

$$\sigma_1(A_1 \Vdash B_1), \dots, \sigma_n(A_n \Vdash B_n) / \sigma(A \Vdash B)$$

holds for L_M iff

$$\sigma_1(A_1 \Rightarrow B_1) \wedge \dots \wedge \sigma_n(A_n \Rightarrow B_n) \supset \sigma(A \Rightarrow B)$$

is a thesis of L_C

Proposition 5.2. Some Boolean correspondences between pragmatic logics and conditional logics.

K corresponds to the system CK+ID of (Chellas, 1975).

P corresponds to the system S_1 of (Burgess 1981), the system P of (Veltman, 1985) and to the system called C in (Bell, 1990a).

R corresponds to the systems V and VC of (Lewis, 1973), to the systems V and $V+MP+CS$ of (Veltman, 1985), and to the system NP of (Delgrande, 1987).

E corresponds to the system $C2$ of (Stalnaker, 1968), and to the systems $V+CEM$ and $V+CEM+MP+CS$ of (Veltman, 1985).

Proof sketch. To establish the left-to-right halves it is sufficient to show that the rules of the pragmatic logic translate to axioms or rules of the conditional logic. This is straightforward in the case of K and CK+ID. The remaining left-to-right halves follow obviously for Veltman's logics. S_1 , P and C are equivalent. Lewis's systems V and VC are respectively equivalent to Veltman's systems V and $V+MP+CS$. Delgrande's system NP is equivalent to V (Bell, 1990d). Finally, the implication holds in this direction for $V+CEM$, and $C2$ and $V+CEM+MP+CS$ are equivalent.

For the right-to-left halves note first that the axioms CS and MP can be replaced by the axiom $(\text{true} \Rightarrow A) \equiv A$ as the translations of Conjunctive Sufficiency and Weak Conditionalization are derivable in the systems concerned. Clearly this axiom can have no impact on the matter. The right-to-left halves now all follow from Theorem 4.13 as this ensures that the iteration of conditionals in arguments has no effect. \square

Providing a formal solution to the pragmatic problem of conditionals means providing a means of constructing models of the theory. In order to evaluate a conditional $A \Rightarrow B$ at a world i the problem is to determine $f(A, i)$. In effect this means building a particular model. Building particular models is not a task that logic, as traditionally viewed, can help with. However pragmatic logics such as preference logics are realisations of the defining models of their pragmatic logics. (Compare the remark in (Brown and Shoham (1989): "While one might imagine that truth maintenance is a computational realisation of

the *proof theory* of some logic, it appears, in fact, to be a realisation of its model theory".) This suggests that suitable pragmatic logics can be developed and used to build models for theories of conditionals. This has been done in (Bell, 1988) where extended causal theories (Bell, to appear) are used to build models for a theory of predictive conditionals.

6. Elaborating the hierarchy

We know from Tarski and Scott (Scott, 1971) that any consequence relation satisfying Reflexivity, Cut and Monotony is the consequence relation of a monotonic logic. Gabbay (Gabbay, 1985) has suggested that the consequence relation of a nonmonotonic logic should satisfy Reflexivity, Cut and Cautious Monotony. Using the results of Section 4, various well known nonmonotonic logics can now be located in a hierarchy on the basis of the properties of their consequence relations. All of the logics considered here are *K*-logics. So Right Weakening is added to Gabbay's conditions. However Cut and CM are replaced by the logically weaker And (Kraus, Lehmann and Magidor, 1990). "Consequence relations" which do not satisfy And can be seen as characterising some form of probabilistic or prelogical reasoning.

Adding Cut brings us to logics like AEL and DL. Despite the fact that these are among the best known of nonmonotonic logics they do not satisfy Gabbay's minimal criteria. So it seems pertinent, if also impertinent, to ask whether these can be called logics. From the technical point of view Cautious Monotony is desirable as it allows for the possibility of constructive definitions of nonmonotonic inference; such as that given by Gabbay. On the intuitive side there is Brewka's pet shop example (Brewka, 1991).

Adding Cautious Monotony brings us to the cumulative logics of (Kraus, Lehmann and Magidor, 1990). These are the smallest logics containing PC which satisfy Gabbay's requirements. An example of a nonmonotonic logic at this level is Brewka's CDL (Brewka, 1991). Inference via the Closed World Assumption (in the context of Horn Clauses) does not contain PC, but is nonetheless cumulative (Makinson, 1989).

Adding Or now gives us the logic of preference logics and examples such as Circumscription.

Adding Rational Monotony brings us to logics which combine both skeptical and credulous entailment. The corresponding conditional logics at this level are Lewis's logic V and VC, and Delgrande's NP. Gärdenfors shows that VC is slightly weaker than his postulates for theory revision (Gärdenfors, 1990). The relations between theory revision and nonmonotonic logic are investigated in (Makinson and Gärdenfors, to appear). On an intuitive level, the connection between nonmonotonic logic,

conditional logic, and the logic of theory revision can be explained in terms of revision involving minimal change. Revision of a belief state *K* to incorporate *A* is rational only if gratuitous changes are avoided; so *K* should be changed as little as possible given the consistency requirement. Similarly, according to the Ramsey test for conditionals (Ramsey, (1931) as extended by Stalnaker (Stalnaker, 1968) a conditional should be evaluated as follows. First add the antecedent (hypothetically) to your stock of beliefs, then make whatever changes are required to maintain consistency (without modifying the hypothetical belief in the antecedent). Finally, consider whether or not the consequent is then true as a result. If it is the conditional is true. If not, not. In the case of nonmonotonic logics, the relevant *A*-models for the examples given in Section 2 were those which differed minimally, in some sense, from the minimal models.

Adding Excluded Middle obliterates the distinction between the skeptical and choice forms of pragmatic entailment and reduces the distinction between these and the choice form to the vacuous case. An example of a pragmatic logic at this level is LMK. LMK is intended to characterise what an ideal agent knows on introspection if initially it knows only *A*. Halpern and Moses restrict their definition of this notion to *honest* sentences; that is, to those sentences which uniquely characterise a knowledge state. Given that *A* is honest, they show that there is a unique S5 model *M* of *A* such that *A* \Vdash_{LMK} *B* iff *M* \vDash *B*. This model is formed by taking the union of the relevant models of *A*, as defined in example 2.5.1. This is possible as the honesty restriction ensures the epistemic uniqueness of such models; the same sentences are known in each of them. It is therefore equivalent to adding em, and distinguishes LMK from preference logics such as Circumscription. LMK differs importantly in this respect from AEL: whereas, for honest *A*, there is a unique knowledge state corresponding to what I know if I know only *A*, there are typically many belief states corresponding to what I might believe if I believe only *A*. A further example is provided by Shoham's causal theories (Shoham, 1988) and their extension (Bell, to appear). Here additional constraints once again guarantee the epistemic uniqueness of the preferred models of a causal theory, and consequently what is known in the theory is what is known in the union of its preferred models. Epistemic uniqueness is an important property in this case as it ensures that (extended) causal theories can be used to represent causal reasoning without fear of the frame problem (Bell, 1990b).

The hierarchy can now be extended to monotonic logics by adding Monotony.

7. Concluding remarks

Traditionally, following Tarski, the notion of entailment for a logic L is defined relative to the set C of models of L ; A entails B in L if every A -model in C is also a B -model; that is:

$$A \vDash_L B \text{ iff } [A]^M \subseteq [B]^M \text{ for all } M \in C$$

C is assumed to be nonempty. Usually C contains many models; typically an infinite number of them. When this is the case it leads to an asymmetry between those inferences which follow in all models in C , the *logical inferences* of L , and those inferences which follow in a particular model M in C , the *pragmatic inferences* of M ; the logical inferences are a proper subset of the pragmatic inferences. If L is sound, this asymmetry is reflected in its proof theory : there is more information in a particular model of a logic than can be obtained from the logic.

Pragmatic entailment is a specialisation of Tarskian entailment in two ways. First $[A]^M$ is replaced by $[A]_M^M$; A no longer has to be true at all A -models in M but only at the relevant A -models. Second, pragmatic entailment is defined in the unusual and limiting case where C is a singleton.

The move from $[A]^M$ to $[A]_M^M$ accounts for the logical properties of pragmatic logics and their relation to conditional logics; pragmatic entailment is a context-dependent notion, Tarskian entailment is not. Pragmatic logics are logically weaker than monotonic logics such as PC; in particular they lack context-free properties such as monotony. This weakening enables fine distinctions to be drawn among consequence relations. These distinctions are important when it comes to correctly representing the logical properties of concepts such as knowledge, belief and causality.

The fact that C is a singleton adds the pragmatic dimension to pragmatic logics : the logical inferences of a pragmatic logic and the pragmatic inferences of its defining model coincide. The proof theory of a pragmatic logic replicates the pragmatic inferences of its defining model. It is nonetheless sound as C is a singleton.

The combined effect of the two restrictions gives us logics whose inferences are exactly those required to represent the logical and pragmatic components of various kinds of pragmatic reasoning. The result is a class of highly specialised logics which occupy the borderline between logic and pragmatics. As such it is reasonable to refer to them as pragmatic logics.

Acknowledgement

I am grateful to David Makinson for his detailed and influential comments on earlier versions of this paper.

References

- J. Allen (1987) *Natural Language Understanding*, Benjamin Cummings, Menlo Park, California.
- J. Bell (1988) Predictive conditionals and reasoning about the future. Manuscript. Computer Science Department, QMW, University of London, London E1 4NS, England.
- J. Bell (1990a) The logic of nonmonotonicity. *Artificial Intelligence* 41 365-374.
- J. Bell (1990b) Why the frame problem is not a problem, *AI Communications* 3(1) 3-10.
- J. Bell (1990c) Relations between nonmonotonic logic and conditional logic. Manuscript. Computer Science Department, QMW, University of London, London E1 4NS, England.
- J. Bell (1990d) Notes on Delgrande's normality conditional. Manuscript. Computer Science Department, QMW, University of London, London E1 4NS, England.
- J. Bell (to appear) Extended causal theories, to appear in *Artificial Intelligence*.
- G. Brewka (1991) *Nonmonotonic Reasoning*, Cambridge University Press, Cambridge.
- A.L. Brown and Y. Shoham (1989) New results on semantical nonmonotonic reasoning, Proc. 2nd International Workshop on Non-Monotonic Reasoning, in: M. Reinfrank et. al. eds, LNAI 346, Springer, Berlin.
- M. Davis (1980) The mathematics of non-monotonic reasoning, *Artificial Intelligence* 13 73-80.
- J. Doyle (1979) A truth maintenance system, *Artificial Intelligence* 12, 231-272.
- P. Gärdenfors (1990) *Knowledge in Flux : Modelling the dynamics of Epistemic States*, MIT Press, Cambridge, Massachusetts.
- J. Halpern and Y. Moses (1984) Towards a theory of knowledge and ignorance: Preliminary Report, TR RJ 4448 48316, IBM Research Laboratory, San Jose, CA.
- I. Kant (1781) *Critique of Pure Reason*, trans. N. Kemp Smith, Macmillan, London, 1929.
- K. Konolige (1987) On the relation between default theories and autoepistemic logics, *IJCAI*, 394-401.

- K. Konolige (1984) Belief and incompleteness, in: J. Hobbs and R. Moore eds., *Formal Theories of the Commonsense World*, Ablex.
- S. Kraus, D. Lehmann, and M. Magidor (1990) Nonmonotonic reasoning, preferential models and cumulative logics, *Artificial Intelligence* 44 167-207.
- D. Lehmann (1989) What does a conditional knowledge base entail?, in: R. Brachman and H. Levesque eds., *Proc. of the International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, San Mateo, CA.
- S. Levinson (1983) *Pragmatics*, Cambridge University Press, Cambridge.
- D. Lewis (1973) *Counterfactuals*, Blackwell, Oxford.
- D. Makinson (1989) General theory of cumulative inference, Proc. 2nd International Workshop on Non-Monotonic Reasoning, in: M. Reinfrank et. al. eds., LNAI 346, Springer, Berlin.
- D. Makinson (1990) Personal communication.
- D. Makinson and P. Gärdenfors (to appear) Relations between the logic of theory change and nonmonotonic logic. To appear in: A. Führmann and M. Morreau eds., *Proc. of the Konstanz workshop on belief revision*, Springer, Berlin.
- J. McCarthy (1980) Circumscription: a form of non-monotonic reasoning, *Artificial Intelligence* 13 27-39.
- R. Montague (1974) Pragmatics, in: *Formal Philosophy*, ed. R. Thomason, Yale Univ. Press.
- R.C. Moore (1983) Semantical considerations on nonmonotonic logic, IJCAI, 272-279.
- R.C. Moore (1984) Possible-world semantics for autoepistemic logic, Proc. 1st International Workshop on Non-Monotonic Reasoning, New York.
- D. Nute (1984) Conditional Logic, in: D. Gabbay and F. Guenther eds., *Handbook of Philosophical Logic*, Vol. 2, D. Reidel, Dordrecht.
- F. Ramsey (1931) General propositions and causality, in: *Foundations of Mathematics and other Logical Essays*, Harcourt Brace, New York.
- R. Reiter (1978) On closed world databases, in: H. Gallaire and J. Minker eds., *Logic and Databases*, Plenum, NY/London 55-76.
- R. Reiter (1980) A logic for default reasoning, *Artificial Intelligence* 13, 81-132.
- D. Scott (1971) Completeness and axiomatizability in many valued logics, in L. Henkin et. al. eds., *Proceedings of the American Mathematical Society*, Providence, R. I.
- Y. Shoham (1987) Nonmonotonic logics: meaning and utility, IJCAI, 388-393.
- Y. Shoham (1988) *Reasoning About Change*, MIT Press, Cambridge Mass.
- R. Stalnaker (1968) A theory of conditionals, in: N. Rescher ed., *Studies in Logical Theory*, Blackwell, Oxford.
- J. Van Benthem (1988) Semantic parallels in natural language and computation, in: M. Garrido ed., *Logic Colloquium, Granada 1987*, North-Holland, Amsterdam.
- F. Veltman (1985) *Logics for Conditionals*, Ph. D. Thesis, University of Amsterdam.

Metalogic for Knowledge Representation

Antonio Brogi

Dipartimento di Informatica
Università di Pisa

Corso Italia 40 - 56125 Pisa, Italy

Franco Turini

Dipartimento di Informatica
Università di Pisa

Abstract

The expressive power of metalogic is shown to be sufficient for the reconstruction of a number of knowledge representation techniques and inference methods. In particular, a metainterpretive definition is given to a number of mechanisms for structuring logic programs, which have been designed to support forms of hypothetical reasoning, mechanisms for the object-orientation of logic theories and hierarchical representation links between theories. The merits of the approach are shown to be on one hand the definition of a unifying framework for comparing different mechanisms and techniques and on the other hand the proof that sophisticated knowledge representation problems can be solved without departing from the classic logic programming framework.

1 INTRODUCTION

Since a procedural interpretation of a subset of first order predicate calculus has been defined (van Emden and Kowalski 1976), several efforts have been devoted to broaden the application area of logic programming. This trend has generated a notable amount of proposals for extending the original logic programming framework to face various problems typical of Artificial Intelligence applications, such as knowledge representation, default reasoning, object-oriented programming and other forms of reasoning. In this paper, we will concentrate on two main classes of extensions proposing mechanisms for structuring and object-orienting logic programs, respectively. Both these kinds of extensions are based on multi-theory frameworks, where knowledge is represented by a set of separate theories and the computation evolves by their composition and interaction. The semantic structures which have been chosen for characterizing these extensions - e.g. Kripke models (Miller 1989) or pos-

sible worlds semantics (Monteiro and Porto 1989) - depart from the simple model-theory of logic programming (van Emden and Kowalski 1976). Moreover, the adoption of different semantic frameworks complicates the comparison of different proposals. In (Brogi et al. 1990a) the definition of a unified framework for structuring logic programs has been presented. If, on one hand, several different extensions have been classified in terms of such a general framework, on the other hand, its semantics has been defined as an extension of the classical one.

In this work, we want to focus on what we believe is a crucial methodological criterium for looking at extensions of logic programming, that is to understand whether their semantics can be expressed within the standard Horn Clause Logic framework or not. The major contribution of the paper is to demonstrate that several extensions of logic programming fall in the first class. The key idea is to use metalogic for the reconstruction of the definition of these extensions according to the amalgamation scheme proposed by (Kowalski 1979, Bowen and Kowalski 1982). The metalevel definition merely consists of definite Horn clauses and thus a classical semantic characterization is entailed. Then, given two extensions, ext_1 and ext_2 say, once we have axiomatized their metalogical provability relations, demo_{ext_1} and demo_{ext_2} , respectively, we can compare them within the classical semantic frameworks (SLD resolution, minimal Herbrand model, fixpoint semantics).

In order to make the reconstructions trustworthy it is necessary to provide evidence that they are equivalent to the original proposals. There are at least four levels on which the evidence can be provided. First of all, we claim that the metainterpretive definitions are so simple and straightforward that the comparison is directly supported by intuition. Second of all, it is possible to put at work our definitions on classic examples on which the other proposals are usually tried and to check that the behaviour is the same. This will be done in the paper, for instance, for analysing our reconstruction of contextual logic programming. Third

of all, one could imagine to show the equivalence on a model theoretic ground. Unfortunately, this is not easy in general, given that different semantic structures are used in the different proposals. At last, the comparison can be carried on through the analysis of the respective operational behaviours. This comparison is facilitated by the fact that most of the considered proposals are given operational semantics in terms of extensions of SLD resolution.

After introducing metalogic in the next section, a set of proposals of mechanisms for structuring logic programs are discussed. In particular, the hypothetical reasoning framework proposed in (Miller 1989) and the contextual logic programming scheme (Monteiro and Porto 1989), together with some of its extensions, are examined. Then some mechanisms for the object-orientation of logic theories are discussed. Hierarchical representation links between theories are considered, also with reference to the knowledge based system Epsilon (Coscia et al. 1988).

2 METALOGIC

We will use the amalgamated logic of (Kowalski 1979, Bowen and Kowalski 1982) where object level expressions are represented by metalevel terms and object level provability is defined by metalevel axioms. Metareasoning has been the subject of several research efforts so far, and is considered today one of the fundamental areas of computational logic (Kowalski 1990). Actually, several applications have been addressed by means of metalogic such as metainterpreters (Sterling and Beer 1989), knowledge assimilators (Bowen and Kowalski 1982), theory consistency and integrity checking (Bry 1990), as well as other forms of reasoning, such as the representation of knowledge and belief (Bowen 1985, Kim and Kowalski 1990). One of the simplest applications of metalogic is the so-called *vanilla* metainterpreter (Hill and Lloyd 1989):

```
demo(T,true)      ←
demo(T,(G1,G2)) ← demo(T,G1),
                  demo(T,G2)
demo(T,G)        ← clause(T,G ← B),
                  demo(T,B)
```

The *vanilla* metainterpreter works for variable-free programs as well as for programs containing variables. Nevertheless, it is worth observing that object level variables are incorrectly represented by metalevel variables. An extensive discussion of the correct representation, where object-level variables are represented by variable-free terms, can be found in (Bowen and Kowalski 1982, Hill and Lloyd 1989, Burt et al. 1990, Kowalski 1990).

As far as a single theory is concerned, the theory argument can be omitted from the *vanilla* metainterpreter.

In moving from a single theory towards a multi-theory framework, where many logic programs can be dynamically combined to form a more complex program, the theory argument becomes relevant. In (Brogi et al. 1990b) a set of theory composition operators has been defined together with a suitable extension of the `demo` predicate to allow the first argument (T) being any term from the algebra devised by a set of constant theory names and a set of theory composition operators. The metalevel definition of two of these operators - `union` and `intersection` of theories - which will be exploited later on, is inductively given on the definition of the `clause` predicate:

```
clause(P ∪ Q, H ← B)      ← clause(P, H ← B)
clause(P ∩ Q, H ← B)      ← clause(Q, H ← B)
clause(P ∩ Q, H ← (B1, B2)) ← clause(P, H ← B1),
                                clause(Q, H ← B2)
```

Theories are named by constant names, while the theory composition operators (`union`, `intersection`) are represented by (infixed) "isomorphic" terms (\cup , \cap). Sometimes in the examples, we use un-named theories which are represented as sets of clauses.

3 STRUCTURING LOGIC THEORIES

3.1 HYPOTHETICAL REASONING

There are several proposals of extensions of logic programming with constructs for the definition of separate theories and their dynamic combination to perform forms of hypothetical reasoning (Gabbay and Reyle 1984, McCarty 1988, Miller 1989). In (Miller 1989) in order to support hypothetical reasoning, implications are allowed in goals and in the bodies of the clauses. An *implication goal* $W \Rightarrow G$ in a theory T (where W is a theory name) is provable if G is provable in the theory T extended with W , actually in $T \cup W$. Operationally, this corresponds to load the code of W before trying to prove the goal G and then discarding it after G succeeds or fails. In (Miller 1989) the semantics of the extension, including also other mechanisms, is given in terms of Kripke models.

This hypothetical reasoning scheme can be provided with a metalogical definition by exploiting the `union` (\cup) operator on theories introduced in the previous section. The following clause has to be added to the definition of the `demo` predicate of section 2:

```
demo(T, W ⇒ G) ← demo(T ∪ W, G)
```

Thus a definition of the hypothetical reasoning scheme of (Miller 1989) can be given in terms of definite Horn clauses only.

The application of metalogic to theory construction supports other forms of hypothetical reasoning (Brogi

et al. 1990b). The vanilla metainterpreter can be exploited to prove goals with the form:

```
demo(T, G)
```

where T can be any (possibly non-ground) term built on theory names with the composition operators. It is worth observing that the theory expression T must not necessarily be ground, that is it may contain free variables.

Some forms of hypothetical reasoning can be reconstructed following this remark. Suppose one aims at determining a set of hypotheses which, when added to a given knowledge base kb , allows to prove a given goal g . In this simple framework, viewpoints can be encapsulated into separate theories, to be possibly composed by means of theory-valued operators. This task can be accomplished by a metalevel goal of the kind:

```
demo(kb ∪ X , g)
```

which will possibly succeed by binding X to an extension of kb where g is provable. For instance, in the oversimplified situation of three theories:

$t_1: \quad t_2: \quad t_3:$

$p \leftarrow q \quad q \leftarrow \quad r \leftarrow$

represented in the clause notation as follows:

```
clause(t1, p ← q) ←
clause(t2, q ← ) ←
clause(t3, r ← ) ←
```

it is possible to determine a theory that, when added to t_1 , allows one to prove the goal p , by means of the goal

```
demo(t1 ∪ X , p)
```

which succeeds, among other possibilities, with $X=t_2$. More generally, the goal $demo(X , p)$, actually constructs (compositions of) theories where p is provable, and succeeds, among other possibilities, with $X=t_1 \cup t_2$.

When dealing with the construction of hypotheses, the ability of generating only non-redundant ones is desired. In the previous example, along with the expected binding for X ($t_1 \cup t_2$), also other redundant solutions would be generated such as: $t_1 \cup t_1 \cup t_2$ or $t_1 \cup t_1 \cup t_1 \cup t_2$ and so on. Methods for enforcing non-redundancy conditions on the hypothesis generation process can be investigated. A first idea is to define some sort of subsumption with the intent of avoiding the generation of redundant terms via a syntactic check. More sophisticated forms of semantic simplifications could be studied as well.

Another interesting direction is to further explore the dual nature of the union and intersection operators under the light of hypothetical reasoning. On one hand, union provides basic means for determining the conditions under which a goal becomes provable in the corresponding theory expression. On the other hand, intersection allows one to determine the conditions under which a goal is still provable in the corresponding theory expression.

A concrete example of application of hypothetical reasoning to fault diagnosis is discussed in (Kowalski 1990). Consider a theory **bicycle** containing some of the causes of bicycle faults:

bicycle:

```
wobbly_wheel ← flat_tyre
wobbly_wheel ← broken_spokes
flat_tyre   ← punctured_tube
flat_tyre   ← leaky_valve
```

To determine the possible causes of **wobbly_wheel**, we can try to extend the knowledge defined by theory **bicycle** with different hypotheses, represented as theories. For instance, by assuming the cause **leaky_valve** true, we are able to prove the goal:

```
demo(bicycle ∪ {leaky_valve ←} , wobbly_wheel)
```

where the union operator is exploited to combine the **bicycle** theory with a specific viewpoint.

In (Kowalski 1990) the particular form of hypothetical reasoning which is dealt with is abduction. In order to determine the possible causes of **wobbly_wheel**, it suffices to seek the abductive explanations of **wobbly_wheel**. Roughly, given a theory T and a possible conclusion C , an abductive explanation of C is defined as a set of sentences Δ such that $T \cup \Delta$ logically implies C . This abductive scheme corresponds to automatically generating hypotheses rather than trying different viewpoints by hand.

In order to achieve the same behaviour in our metalogic scheme, we can represent each abducible predicate with a separate theory. By doing this, in order to find the possible explanations for **wobbly_wheel** it suffices to prove the goal:

```
demo(bicycle ∪ X , wobbly_wheel)
```

where X will be suitably instantiated as discussed above.

3.2 CONTEXTUAL LOGIC PROGRAMMING

An extension of the logic programming paradigm which has been recently proposed is the notion of *Contextual Logic Programming* (Monteiro and Porto 1989). The basic idea is to consider theories as sets of context-

dependent predicate definitions and the proof of a certain kind of formulae (called *extension formulae*) requires extending the context of the proof with the definitions contained in a given theory. The effect of an extension formula $T \gg G$, where T is a theory name and G a goal, is to combine T with the current context C in the following way. According to (Monteiro and Porto 1989), let $C \vdash_c G$ denote that there exists a top-down derivation for a goal G in the context C . Then the effect of an extension formula can be described by the inference rule:

$$\frac{T.C \vdash_c G}{C \vdash_c T \gg G}$$

where the infix'd' is used as the constructor of a stack of theories. The clause to be applied to an atomic goal G in a context ξ can belong either to the most external theory T in the context (i.e. $\xi = T.C$) or to any theory T within the context (i.e. $\xi = C.T.C'$) provided that no definitions for G can be found in the theories preceding T in the context (i.e. in C). Formally:

$$\frac{\begin{array}{l} H \leftarrow B \in T, \vartheta = mgu(G, H), \\ \text{name}(G) \notin C, T.C' \vdash_c (B)\vartheta \end{array}}{C.T.C' \vdash_c G}$$

where `name(G)` stands for the predicate name of the atomic formula G .

Example

Let us consider the following example of derivation, taken from (Monteiro and Porto 1989). There are two theories: one defining what an author is and the other containing information about writers of books.

`authors`:

`author(Person) ← wrote(Person, Sth)`

`books`:

`wrote(plato, republic) ←`
`wrote(homer, iliad) ←`
`author(Person) ← authors >> author(Person)`

The top-derivation (\vdash_c) of the goal `author(plato)` in the context composed of the theory `books`, according to the two inference rules given above, is:

`books $\vdash_c \text{author(plato)}$`

`books $\vdash_c \text{authors} \gg \text{author(plato)}$`

`authors.books $\vdash_c \text{author(plato)}$`

`authors.books $\vdash_c \text{wrote(plato, Sth)}$`

`books $\vdash_c \text{wrote(plato, Sth)}$`

`books $\vdash_c \square$`

□

In (Monteiro and Porto 1989) a possible-worlds declarative semantics is presented, where the contexts (situations) are seen as possible worlds, and the theories as transitions between worlds. A much simpler definition can be given by means of metalogic, extending the `demo` definition of section 2 with the clauses:

```
demo(C, T >> G) ← demo(T ctx C, G)
demo(T ctx C, G) ← clause(T, G ← B),
                     demo(T ctx C, G)
demo(T ctx C, G) ← undefined(G, T),
                     demo(C, G)
```

where the `ctx` functor is the constructor of the stack. It is worth observing that the `undefined` predicate, which describes the overriding mechanism, does not require the handling of negation. Actually, the overriding test can be expressed as a membership test on a finite set of elements (i.e. the names of predicates defined in a theory) so that the `undefined` predicate is defined to succeed when the test fails.

Example (contd.)

The SLD derivation with the `demo` predicate of the goal of the example is then:

```
vdash demo(books, author(plato))
vdash clause(books, author(plato) ← B),
             demo(books, B)
vdash demo(books, authors >> author(plato))
vdash demo(authors ctx books, author(plato))
vdash clause(authors, author(plato) ← B'),
             demo(authors ctx books, B')
vdash demo(authors ctx books, wrote(plato, Sth))
vdash demo(books, wrote(plato, Sth))
vdash □
```

□

3.3 OTHER FORMS OF CONTEXTUAL LOGIC PROGRAMMING

The two basic concepts underlying contextual logic programming are:

- The *context handling mechanism* concerning how the context evolves during the derivation.

- The *composition of predicate definitions* stating which are the clauses defining a predicate when several clauses are composed together.

In the definition presented in (Monteiro and Porto 1989), these two choices are fixed as shown in the inference rule for clause application, given in the previous section:

$$\frac{H \leftarrow B \in T, \vartheta = \text{mgu}(G, H), \\ \underline{\text{name}(G) \notin C, T.C' \vdash_c (B)\vartheta}}{C.T.C' \vdash_c G}$$

By considering a more general scheme for contextual logic programming, the two conditions:

$\text{name}(G) \notin C, T.C' \vdash_c (B)\vartheta$ can be provided of different interpretations. In (Lamma et al. 1989) some extensions to the basic contextual logic programming framework have been defined this way. First, the context to be used in the rest of the derivation can be the whole initial context and not a subpart. Such a context handling mechanism is called *evolving* in (Brogi et al. 1990a).

$$\frac{H \leftarrow B \in T, \vartheta = \text{mgu}(G, H), \\ \underline{\text{name}(G) \notin C, C.T.C' \vdash_c (B)\vartheta}}{C.T.C' \vdash_c G}$$

In order to model such a behaviour, the metalogic definition has to be suitably modified as follows:

```
demo(C, T >> G) ← demo(T ctx C, G)
clause(T ctx C, H ← B) ← clause(T, H ← B)
clause(T ctx C, H ← B) ← undefined(H, T),
                           clause(C, H ← B)
```

A common feature of both the basic contextual logic programming framework and this extension is the overriding interpretation of the theory composition operation. Instead of this overriding mechanism, in (Lamma et al. 1989) a *theory extension* mechanism has been proposed where the condition $\text{name}(G) \notin C$ is omitted. In other words, the choice of which theory to select within a given context is fully nondeterministic. The corresponding change in the metaprograms is trivial: the tests *undefined* are simply omitted.

These different interpretations of the the context handling mechanism and of the composition of predicate definitions, when combined together, offer great expressive power. In (Brogi et al. 1990a) a unified framework for structuring logic programs has been defined by means of these extensions and it is shown how several existing proposals can be classified in terms of the proposed unified scheme, so that several different reasoning methods and various aspects of programming are covered. The scheme is also semantically characterized by defining an extension of the classical Herbrand model semantics of logic programming. It

should be now clear that those non-classical results can be reformulated in terms of the classical semantics via metalogic. For instance, the hypothetical reasoning scheme proposed in (Miller 1989) can be shown equivalent to an instance of the general scheme of contextual logic programming, where a theory extension policy and an evolving context handling mechanism are adopted. Such a correspondence can then be proved directly by comparing the corresponding metainterpreters, as shown in section 5.

4 OBJECT-ORIENTING LOGIC THEORIES

4.1 HIERARCHICAL REASONING

In (Brogi et al. 1990b) some hierarchical operators between logic theories have been defined in terms of the previously mentioned operators of union and intersection of logic theories.

The first link, called *isa*, between a theory P and a theory Q intensionally defines a new theory which contains all the predicate definitions of P and inherits the definitions of predicates occurring in Q only. Notice that the definitions in Q of predicates defined in both theories are replaced by the corresponding definitions in P. A metadefinition of *isa* can now be given as follows:

```
clause(P isa Q, H ← B) ← clause(P, H ← B)
clause(P isa Q, H ← B) ← undefined(H, P),
                           clause(Q, H ← B)
```

Another hierarchical link, called *constraint* (*constr*), intensionally defines a theory which contains the intersection of the definitions of the theories and inherits the definitions of predicates occurring in Q only. Intuitively, theory P constraints theory Q on the commonly defined predicates. The metadefinition of *constraint* exploits the *intersection* operator:

```
clause(P constr Q, H ← B) ← clause(P ∩ Q, H ← B)
clause(P constr Q, H ← B) ← undefined(H, P),
                           clause(Q, H ← B)
```

Let us give two simple examples of the use of these operators.

Consider the following theory *person*:

person:

```
eats(meat)           ←
eats(fish)           ←
eats(vegetables)    ←
lives_in(X)          ← works_in(X)
age(X)               ← current_year(Y),
                           born_on(W),
                           X=Y-W
```

The fact that *John is a vegetarian working in London* can be described by the following *isa* link:

$$\text{john} = \left\{ \begin{array}{l} \text{eats(vegetables)} \leftarrow \\ \text{works_in(london)} \leftarrow \end{array} \right\} \text{isa person}$$

On the other hand, to describe *adult* people we can exploit the **constraint** operator as follows:

$$\text{adult} = \{\text{age}(X) \leftarrow X \geq 14, X \leq 70\} \text{ constr person}$$

In (Monteiro and Porto 1990) a transformational definition of an *isa* link is given. Their approach is quite similar to ours and, indeed, the defined inheritance relation is identical to the *isa* relation defined above. The example below is taken from (Monteiro and Porto 1990) and it is reworked according to our definition. The classes they consider are *pickpocket* which is an *artist* which is a *person*.

person:

activity(working) ←

artist:

likes(X) ← *activity(X)*

pickpocket:

activity(stealing) ←

The above three theories are represented in our terms by the three clauses:

```
clause(person, activity(working) ←)
clause(artist, likes(X) ← activity(X))
clause(pickpocket, activity(stealing) ←)
```

Now, the proof that *pickpockets like stealing* is given by the following derivation:

```
⊤ demo(pickpocket isa artist isa person,
       likes(stealing))

⊤ clause(pickpocket isa artist isa person,
          likes(stealing) ← B),
   demo(pickpocket isa artist isa person, B)

⊤ clause(artist isa person,
          likes(stealing) ← B),
   demo(pickpocket isa artist isa person, B)

⊤ demo(pickpocket isa artist isa person,
       activity(stealing))
```

⊤ clause(pickpocket isa artist isa person,
 activity(stealing) ← B),
 demo(pickpocket isa artist isa person, B)

⊤ demo(pickpocket isa artist isa person,
 true)

⊤ □

4.2 INHERITANCE LINKS IN THE EPSILON SYSTEM

An example of a knowledge base management system based on multiple theories is the one developed within the EPSILON project (Coscia et al. 1988). In EPSILON, each theory contains a chunk of knowledge and is associated to a specific inference engine. There are mechanisms for defining relations among theories and there are two possible types of inheritance: *open* and *closed*.

With the so-called *isa* link, for instance, a theory P inherits from Q clauses to be evaluated in the environment of P. If the inheritance is *closed* (P *clsisa* Q), only the clauses defining predicates which are not defined in P are inherited, while if the inheritance is *open* (P *opnisa* Q) all the clauses of Q are inherited by P. It is worth observing that these two inheritance links of the EPSILON system can be mapped onto some of the previously described operators. More precisely, the *opnisa* link is nothing but the *union* operator defined above, while the *clsisa* corresponds to the *isa* operator of section 4.1:

```
demo(P opnisa Q, G) ← demo(P ∪ Q, G)
demo(P clsisa Q, G) ← demo(P isa Q, G)
```

Another inheritance link defined in EPSILON is the *consultance* link. A theory P, which is linked to another theory Q via a *consultance* link, sends queries to the metainterpreter of Q in an object-oriented fashion. If the link is *closed* (P *clsconsult* Q) a query is sent only for those predicates which are not defined in P, if it is *open* (P *opnconsult* Q) for each predicate. Again, these operators can be fully expressed by means of the previously defined ones. Actually, the *clsconsult* link corresponds to the basic contextual logic programming scheme of section 3.2. The difference between *opnconsult* and *opnisa* only concerns implementation matters, that is which is the actual inference engine which proves the goal. As far as their logical definition is concerned, the two links are semantically equivalent.

```
demo(P opnconsult Q, G) ← demo(P ∪ Q, G)
demo(P clsconsult Q, G) ← demo(P ctx Q, G)
```

4.3 THEORIES AS OBJECTS

Metalevel reasoning in a multi-theory framework naturally parallels some typical aspects of object-oriented programming. A natural object-orientation of a multi-theory framework is to consider each single theory as an object. Objects can exchange messages each other, for instance a theory T_i might send a message Msg to another theory T_j :

T_i :

```
... ←  $T_j * Msg$ 
...
```

Such a message passing primitive can be interpreted by a metalevel call of the form:

$\text{demo}(T_j, Msg)$

where unification provides a bidirectional communication mechanism. In other words, a metalevel call of the form $\text{demo}(T_j, G)$ invoked by some theory T_i can be interpreted as a message sent from T_i to T_j , whose answer will consist of the answer substitution possibly computed by T_j .

So far, we have defined some hierarchical operators between logic theories to be explicitly composed in order to prove some assertions (e.g. $\text{demo}(P \text{ isa } Q, G)$). In most object-oriented systems however, when trying to prove a certain property of an object, there is no need of specifying in the query all the structure of the hierarchy including the object. The inheritance mechanisms are automatically triggered during the computation in order to exploit the hierarchy. Such a behaviour can be obtained within a metalogical framework by representing links between objects with axioms with the form:

$T \text{ link } P \leftarrow$

and suitably extending definition of demo given in section 2 with the following two clauses:

```
demo(T, G)      ← T link P,
                  demo(T link P, G)
demo(T, P * G) ← demo(P, G)
```

If we consider, for example, the definition of the *isa* operator, it is immediate to see that also nested hierarchies (e.g $P \text{ isa } (Q \text{ isa } R)$) can be captured this way.

5 COMPARING DIFFERENT EXTENSIONS

One of the most fascinating aspects of the definition of several different extensions within a unified metalevel

framework is the possibility of performing formal comparisons. Given two extensions, ext_1 and ext_2 say, once we have axiomatized their metalogical provability relations, demo_{ext_1} and demo_{ext_2} , respectively, we can compare them within the classical semantic frameworks (SLD resolution, minimal Herbrand model, fix-point semantics). This way, proposals which are presented in different frameworks can be shown to be equivalent, as well as differences can be formally clarified.

For example, it is interesting to compare some of the proposals for structuring logic programs described in section 3. Let us denote by demo_{hyp} the metaprogram defining the hypothetical reasoning scheme of section 3.1, and demo_{ctx} the one defining the contextual logic programming scheme of section 3.3, where both an evolving context handling mechanism and a theory evolution policy are adopted.

Proposition 5.1

Given two theories P, Q and an atomic goal G :

$$\vdash \text{demo}_{hyp} (P, Q \Rightarrow G) \Leftrightarrow \vdash \text{demo}_{ctx} (P, Q \gg G)$$

Proof

We recall that the two metainterpreters share the clauses:

```
demo(T, true)      ←
demo(T, (G1, G2)) ← demo(T, G1),
                     demo(T, G2)
demo(T, G)         ← clause(T, G ← B),
                     demo(T, B)
```

Then demo_{hyp} contains :

```
demo(T, W ⇒ G)      ← demo(T ∪ W, G)
clause(P ∪ Q, H ← B) ← clause(P, H ← B)
clause(P ∪ Q, H ← B) ← clause(Q, H ← B)
```

while demo_{ctx} contains:

```
demo(C, T ≫ G)      ← demo(T ctx C, G)
clause(T ctx C, H ← B) ← clause(T, H ← B)
clause(T ctx C, H ← B) ← clause(C, H ← B)
```

It is trivial to observe that the two definitions are identical up to renaming of \cup by ctx and of \Rightarrow by \gg . \square

In (Brogi et al. 1990b) the two hierarchical operators defined in section 4.1 have been compared by showing that the success set of P constraint Q is included into the success set of P *isa* Q . Let us see how the same result can be proved by means of the metadefinition of section 4.1.

Proposition 5.2

Given two theories P, Q and an atomic goal G:

$$\vdash \text{demo}(P \text{ constr } Q, G) \Rightarrow \vdash \text{demo}(P \text{ isa } Q, G)$$

Proof

By unfolding the two `demo` definitions we obtain:

$$\begin{aligned} \text{demo}(P \text{ constr } Q, G) &\leftarrow \text{clause}(P, G \leftarrow B_1), \\ &\quad \text{clause}(Q, G \leftarrow B_2), \\ &\quad \text{demo}(P \text{ constr } Q, B_1), \\ &\quad \text{demo}(P \text{ constr } Q, B_2) \end{aligned}$$

$$\begin{aligned} \text{demo}(P \text{ constr } Q, G) &\leftarrow \text{undefined}(G, P), \\ &\quad \text{clause}(Q, G \leftarrow B), \\ &\quad \text{demo}(P \text{ constr } Q, B) \end{aligned}$$

$$\begin{aligned} \text{demo}(P \text{ isa } Q, G) &\leftarrow \text{clause}(P, G \leftarrow B), \\ &\quad \text{demo}(P \text{ isa } Q, B) \end{aligned}$$

$$\begin{aligned} \text{demo}(P \text{ isa } Q, G) &\leftarrow \text{undefined}(G, P), \\ &\quad \text{clause}(Q, G \leftarrow B), \\ &\quad \text{demo}(P \text{ isa } Q, B) \end{aligned}$$

By renaming `constr` by `isa`, the implication is obvious. \square

The examples above highlight a basic proof technology which consists in comparing the metainterpreters by applying straightforward program manipulation techniques. The process is facilitated by the modular structure of the metainterpreters which consists of the vanilla metainterpreter and clauses which capture the theory composition operators.

6 CONCLUSIONS

The use of metalogic is exploited for reconstructing several extensions of logic programming for structuring programs, hypothetical reasoning and object-oriented programming. The results of this investigation show that these extensions can be kept within the classical framework by expressing them by means of metaprograms composed of definite Horn clauses only. The paper advocates, as a by-product, the use of metalogic as a powerful mechanism for programming and reasoning about programs in logic.

Acknowledgments

We would like to thank Paolo Mancarella and Dino Pedreschi for the many valuable discussions on the subject of the paper. This work has been partially supported by Esprit Project 3012 COMPULOG and by the C.N.R. "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo".

References

K.A. Bowen. (1985) Meta-level programming and knowledge representation. *New Generation Computing*,

3:359–383.

K.A. Bowen and R.A. Kowalski. (1982) Amalgamating Language and Metalanguage in Logic Programming. In K.L. Clark and S.A. Tarnlund, editors, *Logic Programming*, pages 153–173. Academic Press.

A. Brogi, E. Lamma, and P. Mello. (1990a) A general framework for structuring logic programs. Technical Report 4/1, CNR - Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Sottoprogetto 4.

A. Brogi, P. Mancarella, D. Pedreschi, and F. Turini. (1990b) Composition Operators for Logic Theories. In J.W. Lloyd, editor, *Computational Logic, Symposium Proceedings*, pages 117–134. Springer-Verlag, Brussels.

Francois Bry. (1990) Intensional Updates: Abduction via Deduction. In D.H.D. Warren and P. Szeredi, editors, *Proc. Seventh International Conference on Logic Programming*, pages 561–578. The MIT Press.

A.D. Burt, P.M. Hill, and J.W. Lloyd. (1990) Preliminary report on the logic programming language Gödel. Technical Report TR-90-02, University of Bristol.

P. Coscia, P. Franceschi, G. Levi, G. Sardu, and L. Torre. (1988) Metalevel definition and compilation of inference engines in the Epsilon logic programming environment. In R. A. Kowalski and K. A. Bowen, editors, *Proc. Fifth International Conference on Logic Programming*, pages 359–373. The MIT Press.

M.H. van Emden and R.A. Kowalski. (1976) The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742.

D.M. Gabbay and N. Reyle. (1984) N-prolog: an extension of prolog with hypothetical implications. *Journal of Logic Programming*, 4:319–355.

P.M. Hill and J.W. Lloyd. (1989) Analysis of metaprograms. In H.D. Abramson and M.H. Rogers, editors, *Metaprogramming in Logic Programming*, pages 23–52. The MIT Press.

J.S. Kim and R.A. Kowalski. (1990) An application of amalgamated logic to multi-agent belief. In *Proceedings of Second Workshop on Meta-Programming in Logic*, pages 272–286.

R.A. Kowalski. (1979) *Logic for problem solving*. North-Holland.

R.A. Kowalski. (1990) Problems and promises of computational logic. In *Proceedings of Symposium on Computational Logic*, pages 1–36. Springer-Verlag.

E. Lamma, P. Mello, and A. Natali. (1989) The design of an abstract machine for efficient implementation of contexts in logic programming. In G. Levi and M. Martelli, editors, *Proc. Sixth International Conference on Logic Programming*, pages 303–317. The MIT Press.

L.T. McCarty. (1988) Clausal intuitionistic logic 1: fixed point semantics. *Journal of Logic Programming*, 5:1–31.

D. Miller. (1989) A logical analysis of modules in logic programming. *Journal of Logic Programming*, 6:79–108.

L. Monteiro and A. Porto. (1989) Contextual logic programming. In G. Levi and M. Martelli, editors, *Proc. Sixth International Conference on Logic Programming*, pages 284–302. The MIT Press.

L. Monteiro and A. Porto. (1990) A Transformational View of Inheritance in Logic Programming. In D.H.D. Warren and P. Szeredi, editors, *Proc. Seventh International Conference on Logic Programming*, pages 481–494. The MIT Press.

L. Sterling and R.D. Beer. (1989) Metainterpreters for expert systems construction. *Journal of Logic Programming*, 6:163–178.

The Monotonic Abduction Problem: A Functional Characterization on the Edge of Tractability

Tom Bylander

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210
USA

Abstract

In the monotonic abduction problem, a composite hypothesis always explains at least as much data as any of its subsets. Previous work has shown that finding an explanation for a monotonic abduction instance is tractable, but enumerating explanations and finding a best explanation are intractable. This paper extends these results in two ways. One, I present a tractable algorithm for finding bounds on a best explanation and analyze the properties of these bounds. Two, I show that a wide variety of proposed representations for abduction are functionally similar to the monotonic abduction problem; that is, each representation under suitable restrictions is subject to the complexity results for the monotonic abduction problem.

1 INTRODUCTION

Abduction can be characterized as finding the best explanation for a set of data. If explanations are formed out of combinations of hypotheses, then abduction can be thought of as finding the best composite hypothesis that explains all the data (Josephson et al., 1987).

Bylander et al. (1989, 1990) considered several problems of this type, in each case, demonstrating for each problem whether finding the best explanation is tractable or not.¹ In particular, the *monotonic abduction problem* (Allemang et al., 1987) has both tractable and intractable aspects. In this problem, a composite hypothesis always explains at least as much data as any of its subsets. Previous work has shown that it is tractable to find an explanation, but intractable (NP-hard) to enumerate explanations and to find the best explanation (Bylander et al., 1990).

One way to “address” intractability is to look for tractable subproblems of the intractable problem, examples of which are described in Bylander et al. (1989, 1990). However, it is unrealistic to expect that real-world problems always belong to some tractable problem. For such cases, it is useful to come as close to the best explanation as can be done efficiently. One main result of this paper is an algorithm that finds two sets of hypotheses that are guaranteed to bound a best explanation for the monotonic abduction problem.

A difficulty with this result as well as the previous results is that the monotonic abduction problem is defined using *subfunctions* rather than *representations*. I assume the existence of tractable functions from composite hypotheses to the data they explain and from composite hypotheses to their relative plausibilities, and define the monotonic abduction problem by a constraint on the explanation function. The power of such an analysis is that the results applies to any representation of abduction problems that is functionally equivalent. Unfortunately, other representations have similarities, but are not equivalent. Nevertheless, the other main result describes when representations as diverse as set covering, Horn theories, Bayesian networks, etc. are sufficiently similar to the monotonic abduction problem so that the complexity results apply. In a sense, these representations are variants for encoding essentially the same functions. The invariant is the subfunctional characterization, i.e., the definition of the monotonic abduction problem.

Before presenting my new results, I first describe the model of abduction and previous results for the monotonic abduction problem. Next, I present and analyze our algorithm for finding bounds on a best explanation for the monotonic abduction problem. Finally, I list several abduction representations that are functionally similar to the monotonic abduction problem.

¹In previous papers, “class of problems” and “problem” were used instead of the traditional “problem” and “instance,” respectively. I adopt the latter terminology here.

2 MODEL OF ABDUCTION

Following Bylander et al. (1990), an instance of the *abduction problem* is a tuple $\langle D_{all}, H_{all}, e, pl \rangle$, where:

D_{all} is a finite set of all the data to be explained;

H_{all} is a finite set of all the individual hypotheses; each subset $H \subseteq H_{all}$ corresponds to a *composite hypothesis*, i.e., a hypothesis that each $h \in H$ is present, and each $h \notin H$ is absent or irrelevant;

e is a map from subsets of H_{all} to subsets of D_{all} (H explains $e(H)$); and

pl is a map from subsets of H_{all} to a partially ordered set (H has plausibility $pl(H)$).

For convenience, I define a function that determines which individual hypotheses can contribute to explaining a datum. Although it is not a true inverse, I refer to this function as e^{-1} , formally defined as:

$$e^{-1}(d) = \{h \mid \exists H \subset H_{all} (d \notin e(H) \wedge d \in e(H \cup \{h\}))\}$$

Note that $h \in e^{-1}(d)$ does not imply $d \in e(h)$.

A composite hypotheses H is *complete* if $e(H) = D_{all}$. That is, H explains all the data.

H is *parsimonious* if $\exists H' \subset H (e(H) \subseteq e(H'))$. That is, no proper subset of H explains all the data that H does.

H is an *explanation* if it is complete and parsimonious.

H is a *best explanation* if it is an explanation, and if there is no explanation H' such that $pl(H') > pl(H)$. That is, no other explanation is more plausible than H . It is possible that an abduction instance has no best explanation (because there are no explanations), or that it might have several best explanations due to equal or unordered plausibility values.

Clearly, what counts as a best explanation depends on how plausibilities of composite hypotheses are determined, which in turn depends on belief value dependencies among data and hypotheses. To simplify the analysis, I adopt the following somewhat crude, qualitative criterion for comparing explanations called the *best-small plausibility criterion* (Bylander et al., 1990):

$$\begin{aligned} pl(H) > pl(H') &\leftrightarrow \\ \exists m: H \rightarrow H' \text{ } (m \text{ is 1-1} \wedge \\ \forall h \in H \text{ } (pl(h) \geq pl(m(h))) \wedge \\ (|H| = |H'| \rightarrow \\ \exists h \in H \text{ } (pl(h) > pl(m(h))))) \end{aligned}$$

To be more plausible according to best-small, the elements of H need to be matched to the elements of H' so that the elements of H are at least as plausible as their matches in H' . If H and H' are the same size, then in addition some element in H must be more plausible than its match in H' . Note that if H is larger than H' , then $pl(H) > pl(H')$.

In other words, the best-small criterion prefers more plausible individual hypotheses and smaller composite hypotheses. However, no ordering is imposed if these preferences conflict. For example, a composite hypothesis is not considered better or worse than a smaller composite hypothesis with less plausible elements.

From a probabilistic viewpoint, best-small assumes that individual hypotheses are conditionally independent of each other given the data. Consequently, for probabilistic representations of abduction, it is easy to create abduction instances that do not adhere to best-small. However, for the purposes of this paper, it is sufficient to reduce from finding a best explanation in a best-small abduction problem to finding a best explanation using the abduction representation. I shall take as given that this is possible for any reasonable abduction representation.² Thus, any intractability result for a best-small abduction problem also holds for "reasonable" abduction representations. However, a tractability result for a best-small abduction problem must be carefully interpreted in the context of abduction representations.

3 THE MONOTONIC ABDUCTION PROBLEM

An abduction instance is *monotonic* if (Allemang et al., 1987):

$$\forall H, H' \subseteq H_{all} (H \subseteq H' \rightarrow e(H) \subseteq e(H'))$$

That is, a composite hypothesis explains at least as much data as its subsets. Thus, if a composite hypothesis explains a datum, so does any superset. However, it does not follow that any proper subset of the composite hypothesis explains the datum, i.e., the datum might be explained only by the "interaction" of all the individual hypotheses in the composite hypothesis.

An example of monotonic abduction is model-based diagnosis without fault models (Davis, 1984; Genesereth, 1984; Friedrich et al., 1990). In this style of diagnosis, a functioning component is described by a set of constraints, while a faulty component is described by no constraints. As a result, hypothesizing a faulty component permits a wider range of possible behaviors to be consistent. Thus, as additional components are hypothesized to be faulty, the same or more data are explained, i.e., the monotonic property holds.

Assuming that e and pl are tractable, Bylander et al. (1990) showed the following:

Theorem 1 *For the monotonic abduction problem, there is a tractable algorithm for finding an explanation, if one exists.*

²A reduction to belief revision using Bayesian belief networks (Pearl, 1987) appears in Bylander et al. (1990).

Theorem 2 For the monotonic abduction problem, it is NP-complete to determine whether an additional explanation exists, given a set of explanations.

Theorem 3 For the monotonic abduction problem using the best-small plausibility criterion, it is NP-hard to find a best explanation.

That is, finding one explanation is easy, but enumerating explanations and finding a best explanation are hard. With regard to the last result, the combinatorics arise from choices between individual hypotheses with similar plausibility. These results leave open a significant gap between what can be done tractably and what can be done intractably. While some explanation can be found tractably, Theorem 3 casts considerable doubt about how good it can be. The next section shows how close a best explanation can be achieved with a tractable algorithm.

4 FINDING BOUNDS FOR A BEST EXPLANATION

It turns out that finding non-trivial bounds on a best explanation is tractable for the monotonic abduction problem. Two sets of individual hypotheses L and U are said to bound a best explanation if there exists a best explanation H such that $L \subseteq H \subseteq U$. Finding trivial bounds is easy; \emptyset and H_{all} bound all best explanations. Algorithm 1 finds more interesting bounds.

Algorithm 1 repeatedly finds upper and lower bounds until the same lower bound reoccurs. The idea is to cycle between finding the most plausible hypotheses to explain the data not yet explained (the upper bound), and then, out of this set, selecting those hypotheses that are essential for explaining some datum (the lower bound).

The loop that finds an upper bound adds hypotheses to the current lower bound until a complete composite hypothesis is found. For each hypothesis h not in the current lower bound L , the loop first determines the data D that h contributes to. Then, it determines the hypotheses H that are more plausible than h . If H and L together do not explain D , then h is selected to be part of the next upper bound U . In other words, the selected hypotheses are the most plausible individual hypotheses that contribute to explaining the data not explained by the lower bound.

The loop that finds a lower bound removes hypotheses from the current upper bound that are superfluous. For each hypothesis h in the current upper bound U , the loop tests if $U \setminus \{h\}$ explains all the data. If so, then h is not part of the next lower bound L . That is, the next lower bound only contains those individual hypotheses that cannot be removed from the current upper bound without loss of explanatory power. The

L stands for the current lower bound.
 U stands for the current upper bound.
 W stands for a working hypothesis.

Determine whether an explanation exists.
If $e(H_{all}) \neq D_{all}$ then
 Return nil

Find bounds on a best explanation.

$W \leftarrow \emptyset$
repeat
 Determine upper bound.
 $L \leftarrow W$
 for $h \in H_{all} \setminus L$
 $D \leftarrow \{d \mid h \in e^{-1}(d)\}$
 $H \leftarrow \{h' \mid pl(h') > pl(h)\}$
 if $D \not\subseteq e(H \cup L)$ then
 $W \leftarrow W \cup \{h\}$
 end for
 $U \leftarrow W$

Determine lower bound.
for $h \in U$
 if $D_{all} = e(U \setminus \{h\})$ then
 $W \leftarrow W \setminus \{h\}$
 end for
until $W = L$
return $\langle L, U \rangle$

Algorithm 1: Finding Bounds on a Best Explanation for the Monotonic Abduction Problem Using the Best-Small Plausibility Criterion

lower bound might not be complete because each hypothesis out of a set of hypotheses might be superfluous, but not the set as a whole.

I have demonstrated the following:

Theorem 4 For the monotonic abduction problem using the best-small plausibility criterion, Algorithm 1 finds bounds on a best explanation.

The proof is given in the appendix. To put it simply, this is because each hypothesis in the upper bound is among the most plausible for explaining for some datum, and each hypothesis in the lower bound is demonstrably essential in the context of the most plausible individual hypotheses. Best-small's preference for more plausible individual hypotheses ensures that a best explanation is between the bounds.

Theorem 5 With appropriate caching, the time complexity of Algorithm 1 is $O(n^3 + n^2 C_e + nC_{e-1} + nC_{pl})$, where $n = |D_{all} + H_{all}|$, and pC_f denotes p calls of the function f .

The outer loop iterates at most n times because at

worst one hypothesis at a time is added to the lower bound. The upper bound loop is an n^2 loop because n set comparisons are made, and the values of D and H for individual hypotheses can be cached. The lower bound loop is also an n^2 loop because of set comparisons. Each computation of the upper bound loop calls e up to n times. The lower bound loop also calls e up to n times. Only n calls to e^{-1} and pl are necessary because they are only invoked for individual data and hypotheses, respectively.

The nature of the bounds can be characterized as follows. An individual hypothesis h is *locally best* for a datum d if:

$$\begin{aligned} h \in e^{-1}(d) \wedge \\ \forall H \subseteq H_{all} (d \in e(H) \rightarrow \exists h' \in H (pl(h') \not> pl(h))) \end{aligned}$$

That is, h contributed to explaining d and some hypothesis of similar or lower plausibility is required to explain d . A set of hypotheses is called locally best if each element of the set is locally best for some datum. The upper bound loop in Algorithm 1 initially selects all locally best hypotheses.

An individual hypothesis h is *essential* if:

$$\begin{aligned} \exists H \subseteq H_{all} (e(H) = D_{all}) \wedge \\ \forall H \subseteq H_{all} (e(H) = D_{all} \rightarrow h \in H) \end{aligned}$$

That is, h is needed to fully explain the data. A set of hypotheses is called essential if each element of the set is essential. The lower bound loop in Algorithm 1 would select all essential hypotheses if H_{all} was the current upper bound. Successive lower bounds are supersets of the essential hypotheses, and successive upper bounds are subsets of the locally best hypotheses. The following iterative definitions further refine this characterization:

$$\begin{aligned} local-best_0 &= \{h \mid h \text{ is locally best for some } d \in D_{all}\} \\ essential_0 &= \{h \mid h \text{ is essential}\} \\ local-best_i &= essential_{i-1} \cup \\ &\quad \{h \mid h \in e^{-1}(d) \wedge \\ &\quad \forall H \subseteq H_{all} \setminus essential_{i-1} \\ &\quad (d \in e(H \cup essential_{i-1}) \rightarrow \\ &\quad \exists h' \in H (pl(h') \not> pl(h)))\} \\ essential_i &= \{h \mid \exists H \subseteq local-best_{i-1} \\ &\quad (e(H) = D_{all}) \wedge \\ &\quad \forall H \subseteq local-best_{i-1} \\ &\quad (e(H) = D_{all} \rightarrow h \in H)\} \end{aligned}$$

That is, $local-best_0$ and $essential_0$ are those hypotheses that are locally best and essential, respectively. $local-best_i$ and $essential_i$ are those hypotheses that are, respectively, locally best in context of $essential_{i-1}$ and essential in the context of $local-best_{i-1}$. Algorithm 1 first computes $local-best_0$, then $essential_1$, then $local-best_2$, and so on. In essence, Algorithm 1 repeatedly applies the criteria for locally best and essentialness assuming that non-locally best hypotheses are false, and that essential hypotheses are true.

The lower and upper bounds found by Algorithm 1 can now be precisely characterized. The lower bound is the limit of $essential_i$ as i goes to infinity. The upper bound is the limit of $local-best_i$ as i goes to infinity.

These characterizations can be used to identify cases where Algorithm 1 makes little or no progress. No progress can occur if the essential and locally best properties do not differentiate between hypotheses. For example, no hypothesis is essential if each datum can be separately explained by two hypotheses, and every hypothesis is locally best if all the hypotheses that contribute to explaining a datum have the same plausibility. A less extreme case is if no hypotheses are essential, and if each datum can be separately explained by its two most plausible hypotheses, both of which are equally plausible. The locally best property selects the two best hypotheses for each datum, but because there are no essential hypotheses in this context, no further progress can be made.

Identifying cases where Algorithm 1 converges on a best explanation is a little harder. The most general case I have discovered is if the hypotheses that contribute to each datum can be totally ordered without any ties. In this case, $essential_{i+1}$ must include some hypothesis in addition to $essential_{i-1}$ (if convergence has not yet occurred, i.e., $essential_{i-1} \neq local-best_i$). This is because the least plausible hypothesis h selected by $local-best_i$ must be essential in the context of $local-best_i$. If more plausible hypotheses were sufficient, h would not have been selected. h becomes part of $essential_{i+1}$ because no additional hypotheses of lower plausibility were selected.

It should be mentioned that Algorithm 1 finds bounds for only one of possibly several best explanations, i.e., it finds bounds for one "good" explanation, but many other "good" explanations might exist. However, given a best-small best explanation in a monotonic abduction instance, it turns out to be NP-complete to determine whether a second best explanation exists (Bylander et al., 1990). Thus, Algorithm 1 appears to be the best that can be tractably done for the monotonic abduction problem.

There are a couple of directions in which this analysis could be extended. Most notable is relaxing the assumption made by the best-small plausibility criterion that individual hypotheses are conditionally independent given the data. Of course, a key constraint is to maintain tractability while relaxing this assumption.

Another direction is to consider the complexity of average cases. For example, if the average size of best explanations is small enough, then one would expect that finding *all* best explanations would have average polynomial complexity. One could also consider the average number of hypotheses that contribute to a datum, the average number of hypotheses with the same plausibility, etc.

5 FUNCTIONAL SIMILARITY TO ABDUCTION REPRESENTATIONS

The previous sections defined the monotonic abduction problem and the best-small plausibility criterion and described several complexity results and an algorithm related to finding best explanations. However, I provided no representation for abduction. I simply assumed the existence of appropriate representations for hypotheses, data, and their relationships, and assumed the existence of appropriate methods that used these representations for determining explanatory coverage and plausibility.

The power of this kind of functional analysis is that any abduction representation that is *functionally equivalent* to the monotonic abduction problem is subject to the complexity results for the monotonic abduction problem. Functional equivalence between two formalisms can be demonstrated with polynomial reductions between them. Thus, what is tractable/intractable for the monotonic abduction problem is tractable/intractable for the representation.

However, this kind of functional equivalence turns out to be difficult to prove. Fortunately, a weaker condition, which I call functional similarity, is sufficient for the complexity results to carry over. The remainder of this section illustrates the difficulty and its solution for one abduction representation and lists other functionally similar representations.

5.1 FUNCTIONAL SIMILARITY TO ASSUMPTION-BASED EXPLANATIONS

Consider propositional Horn theories as a representation for abductive knowledge as proposed by Selman and Levesque (1990). Following their paper, a propositional Horn theory is specified by a set of clauses Σ , where each clause is a disjunction of literals (letters or their negations) with at most one positive literal. For convenience, each clause is represented by a set, i.e., $\bar{p} \vee q$ by $\{\bar{p}, q\}$.

Given a letter q , an “explanation” for q is defined to be a minimal set of unit clauses α such that $\Sigma \cup \alpha \models q$, and $\Sigma \cup \alpha$ is consistent. If an explanation for q is restricted to letters from a designated set A , then the explanation is considered an “assumption-based explanation.”

If Σ is restricted to definite Horn clauses, i.e., that each Horn clause contain exactly one positive literal, then the assumption-based explanation problem (Σ, A, q) can be reduced to the monotonic abduction problem:

$$\begin{aligned} D_{all} &= \{q\} \\ H_{all} &= A \\ e(H) &= \begin{cases} \{q\} & \text{if } \Sigma \cup \{\{h\} \mid h \in H\} \models q \\ \emptyset & \text{otherwise} \end{cases} \\ pl(H) &= \text{anything} \end{aligned}$$

The point being that if $\Sigma \cup \alpha \models q$ and $\alpha \subset \alpha'$, then $\Sigma \cup \alpha' \models q$ also. Consistency is not a problem for definite Horn clauses because Σ union any set of positive unit clauses is consistent. In essence, the reduction simply uses the Horn theory for representing the e function.

However, reducing a monotonic abduction problem to an assumption-based explanation problem might require a Horn theory of possibly exponential size. The “natural” way to construct a Horn theory from a monotonic abduction instance is as follows (D_{all} is short for $\overline{d_1, d_2, \dots}$):

$$\begin{aligned} \Sigma &= \{\{D_{all}, q\}\} \cup \\ &\quad \{\{H, d\} \mid H \subseteq H_{all} \wedge d \in D_{all} \wedge d \in e(H) \wedge \\ &\quad \quad \quad \neg H' \subseteq H (d \in e(H'))\} \\ A &= H_{all} \end{aligned}$$

That is, Σ contains a Horn clause $D_{all} \rightarrow q$, and, for each datum d and each minimal set of hypotheses H that explains d , Σ contains a Horn clause $H \rightarrow d$.³ The difficulty is that there might be an exponential number of such minimal sets.

Fortunately, a strong functional equivalence with the monotonic abduction problem is not necessary for the complexity results to apply. It turns out that a subproblem of the monotonic abduction problem called the *independent abduction problem* has very similar properties (Bylander et al., 1990). An independent abduction instance is constrained by:

$$\forall H \subseteq H_{all} (e(H) = \bigcup_{h \in H} e(h))$$

That is, a composite hypothesis explains a datum if and only if one of its elements explains the datum. Every independent abduction instance is a monotonic abduction instance, but the converse is not true.

Of Theorems 1-3, the only one that is not known to also apply to independent abduction problems is Theorem 2, which concerns the complexity of enumerating explanations. Instead, the following somewhat weaker result is known (Bylander et al., 1990):

Theorem 6 *For the independent abduction problem, it is #P-complete to determine the number of explanations.*

³So to explain q from the designated set of assumptions $A = H_{all}$, a minimal set of hypotheses must be chosen that implies D_{all} , which in turn implies q .

That is, determining the number of explanations for the independent abduction problem is just as hard as determining the number of solutions to an NP-complete problem. It is an open question whether enumerating an additional explanation is hard for the independent abduction problem; nevertheless, enumeration of all explanations remains an intractable problem.

Thus, to show a kind of *functional similarity* of a representation to the monotonic abduction problem, two results are sufficient: a reduction from the independent abduction problem to the representation and a reduction from the representation to the monotonic abduction problem. These results are sufficient for Theorems 1, 3, and 6 to apply. Algorithm 1 also applies if the best-small plausibility criterion is appropriate.

An independent abduction problem can be polynomially reduced to a Horn theory.

$$\begin{aligned}\Sigma &= \{\{\overline{D_{all}}, q\}\} \cup \\ &\quad \{\{\overline{h}, d\} \mid h \in H_{all} \wedge d \in D_{all} \wedge d \in e(\{h\})\} \\ A &= H_{all}\end{aligned}$$

The reduction is now polynomial because every minimal set of hypotheses that explains a datum is a singleton set. Therefore, the assumption-based explanation problem restricted to definite Horn clauses is functionally similar to monotonic abduction problems, which leads to the following corollaries:

Corollary 7 *For the assumption-based explanation problem restricted to definite Horn clauses, there is a tractable algorithm for finding an explanation, if one exists.*

Corollary 8 *For the assumption-based explanation problem restricted to definite Horn clauses, it is $\#P$ -complete to determine the number of explanations.*

Corollary 9 *For the assumption-based explanation problem restricted to definite Horn clauses using the best-small plausibility criterion, it is NP-hard to find a best explanation.*

Corollary 10 *For the assumption-based explanation problem restricted to definite Horn clauses using the best-small plausibility criterion, Algorithm 1 finds bounds on a best explanation and is tractable.*

What happens if indefinite Horn clauses are permitted? As one might expect, this problem is harder—Selman and Levesque (1990) show that even finding an explanation is intractable. As it turns out, this result can also be derived using another abduction problem defined in Bylander et al. (1990), the *independent incompatibility abduction problem*. An instance of this problem specifies a set \mathcal{I} , whose elements are two-element subsets of H_{all} , indicating pairs of hypotheses that are incompatible with each other. For this

problem, e is defined as follows:

$$e(H) = \begin{cases} \emptyset & \text{if } \exists I \in \mathcal{I}(I \subseteq H) \\ \bigcup_{h \in H} e(\{h\}) & \text{otherwise} \end{cases}$$

It is easy to represent \mathcal{I} with Horn clauses, e.g., h incompatible with h' can be implemented with the indefinite Horn clause $\{\overline{h}, \overline{h'}\}$.

For the independent incompatibility abduction problem, it is NP-complete to determine if an explanation exists (Bylander et al., 1990). Thus, the capability to express incompatible hypotheses/assumptions “explains” the hardness of the assumption-based explanation problem.

5.2 SIMILARITY TO OTHER ABDUCTION REPRESENTATIONS

Similar results can be derived for virtually every known representation for abduction problems. Due to space and time limitations, I shall simply list a sample. If suitable reductions have appeared elsewhere, a citation is given.

The set covering model of Reggia et al. (1983) can be reduced to the independent abduction problem. Reductions are given in Allemang et al. (1987). The parsimonious covering theory of Peng and Reggia (1987) and the cover-and-differentiate method of Eshelman (1988) also use set covering for determining explanatory coverage, although they compute plausibility differently.

The belief revision theory of Pearl (1987) if Bayesian networks are restricted to two levels (i.e., one level of “causes” directed to one level of “effects”), and the interactions from uninitialized variables (the causes) to initialized variables (the effects) are restricted to noisy-OR and noisy-AND. Reductions are given in (Bylander et al., 1990).

The theory of diagnosis of Reiter (1987) if the use of the AB predicate in the structural description is restricted to formulas of the form $\neg AB(c) \vee \dots$, i.e., AB is only used as a negated disjunct in the premises of formulas. This appears to be equivalent to model-based diagnosis without fault models (Friedrich et al., 1990). Reductions for Reiter’s theory of diagnosis are given in (Bylander et al., 1990).

The use of Theorist for abductive reasoning (Poole, 1989) limited to the following “commands”: conjectures of the form h_i , defaults of the form $h_{i1} \wedge h_{i2} \wedge \dots \Rightarrow d_j$ and observations of the form d_j .

The theory of explanatory coherence (Thagard, 1989) if no incoherence relations are used.

Except for the set covering approaches, restrictions are needed to ensure functional similarity to the mono-

tonic abduction problem. Just as in the assumption-based explanation problem, these representations permit even more difficult abduction problems to be represented. For these representations then, the definition of monotonic abduction problems provides a functional characterization of a major dividing line between the tractable and the intractable.

5.3 REPRESENTATIONAL AND SUBFUNCTIONAL ANALYSES

One approach to analyzing the computational complexity of a reasoning task is *representational analysis*, i.e., propose a representation for the task, propose a truth or belief theory of what conclusions can be inferred from the representation, and consider the complexity of performing a given type of inference with the representation, varying restrictions on the expressiveness of the representation (Levesque and Brachman, 1985). A representational analysis provides an understanding of the restrictions that are required to guarantee tractability on some task, but it requires an *a priori* commitment to both a representation and a belief theory.

My approach follows an alternative approach, that of *subfunctional analysis*. I specify subfunctions that are available for performing the computation (e.g., explanatory coverage and plausibility), use the subfunctions to define constraints on the input and output to a problem (e.g., abduction problem and best explanation), and consider the complexity of performing the computation, varying restrictions on the behavior of the subfunctions. A subfunctional analysis also provides an understanding of the restrictions that guarantee tractability, but it requires different commitments—that the computation depends on certain subfunctions, and that the knowledge for computing those subfunctions is available in some form.

Fundamentally, subfunctional analyses can provide results of greater generality because the same function can be implemented using different representations and belief theories with essentially the same computational properties. This paper provides a striking example of this generality. The results of a subfunctional analysis for a particular abduction problem, the monotonic abduction problem, apply to widely different representations with widely different belief theories, including set covering, probabilistic belief revision (choose the most probable explanation), consistency-based abduction (an explanation must be logically consistent with the data), and implication-based abduction (an explanation must logically imply the data).

Of course, some representation(s) must eventually be chosen to implement a reasoning task, and representational analyses are necessary and important for understanding the computational consequences of these choices. However, it is important to understand what

consequences are independent of representation choice, and subfunctional analyses can serve to provide this understanding.

6 REMARKS

In this paper, I have described the monotonic abduction problem, presented an algorithm that tractably find bounds on a best explanation, and discussed how other proposed representations for abduction problems are functionally similar. It is appropriate to characterize monotonic abduction problems as on the “edge” of tractability because finding non-trivial bounds on a best explanation is tractable, but finding a best explanation is not, much less finding all the best explanations.

It is also appropriate to characterize the definition of monotonic abduction problems as “functional” because it does not make any commitment to a representation of abductive knowledge or a truth or belief theory for interpreting the representation. As can be seen from the literature, several different kinds of representational commitments are possible. But to understand the functions computed from these representations, the definition of the monotonic abduction problem remains a fundamental invariant.

A PROOF OF CORRECTNESS OF ALGORITHM 1

First Algorithm 1 determines if H_{all} is complete, i.e., $e(H_{all}) = D_{all}$. If H_{all} is complete, then some minimal subset H of H_{all} must also be complete, i.e., H is complete and parsimonious, hence H is an explanation. If an explanation exists, then some explanation must be maximally plausible, which means that a best explanation exists. Conversely, if a best explanation H exists, then, by the monotonic property, every superset of H is complete. Consequently, H_{all} must also be complete. Therefore, determining if H_{all} is complete is sufficient to determine if a best explanation exists in a monotonic abduction instance.

Now consider (abstractly) sorting the set of best explanations by their least plausible elements, i.e., first sort by the least plausible element, then sort by the next least plausible element, and so on. Let H_{best} be one of the best explanations ranked highest by this sorting procedure, i.e., H_{best} ’s least plausible elements are the “most plausible” compared to other best explanations. I shall demonstrate that Algorithm 1 bounds H_{best} .

Suppose that L and U bound H_{best} (for the purposes of the proof, I assume that L is initially \emptyset and U is initially H_{all}).

Consider the loop that determines a new upper bound. This loop selects an individual hypothesis h only if

$h \in L$ or if L and the hypotheses more plausible than h are insufficient to explain the data that h contributes to. H_{best} cannot contain an element that does not pass this test because the element could then be replaced with more plausible elements, and, as a result, H_{best} would not have been ranked highest by the sorting procedure above. Thus, it must be that H_{best} only contains elements that pass this test. Consequently, the new value of U must be an upper bound.

Consider the loop that determines a new lower bound. This loop selects an individual hypothesis h only if $U \setminus \{h\}$ is not complete. Because U is an upper bound of H_{best} , H_{best} must contain these hypotheses; otherwise, H_{best} would not be complete and also would not be a best explanation. Thus, the new value for L must be a lower bound.

To demonstrate that Algorithm 1 halts, it is sufficient to show that the new value for L is a superset of the previous value of L . Because L is initially \emptyset , this obviously is true for the first two values of L . Denote the successive values of L as L_1, L_2, L_3, \dots . Let U_i be the value of the upper bound generated using L_i . Suppose $L_{i-1} \subseteq L_i$. Note that in the upper bound loop, the values of D and H depend only on the value of h , not on L or U . Thus, if h is selected for U_i , then $D \not\subseteq e(HUL_i)$. Because $HUL_{i-1} \subseteq HUL_i$ and the monotonic property holds, $D \not\subseteq e(HUL_{i-1})$ either. So h would be selected for U_{i-1} also. That is, $U_i \subseteq U_{i-1}$. Now if h is selected for L_i , then $D_{all} \neq e(U_{i-1} \setminus \{h\})$. This implies that $D_{all} \neq e(U_i \setminus \{h\})$ also, i.e., h is also selected for L_{i+1} . Hence, $L_i \subseteq L_{i+1}$. Therefore, by mathematical induction, each new value for L must be a superset of the previous value.

Because Algorithm 1 maintains bounds on a best explanation, and it halts, Algorithm 1 finds bounds on a best explanation.

Acknowledgments

The ideas in this paper are a result of several discussions with John Josephson and Ashok Goel. As usual, this does not imply complete agreement. This research has been supported in part by the National Heart, Lung and Blood Institute through NIH Grant 1 R01 HL 38776-01 and by the National Institute on Disability and Rehabilitation Research through grants H133C90090 and H133E80017.

References

- Allemang, D., Tanner, M. C., Bylander, T., and Josephson, J. R. (1987). On the computational complexity of hypothesis assembly. In *Proc. Tenth Int. Joint Conf. on Artificial Intelligence*, pages 1112–1117, Milan.
- Bylander, T., Allemang, D., Tanner, M. C., and Josephson, J. R. (1989). Some results concerning the computational complexity of abduction. In *Proc. First Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 44–54, Toronto.
- Bylander, T., Allemang, D., Tanner, M. C., and Josephson, J. R. (1990). The computational complexity of abduction. *Artificial Intelligence*, to appear.
- Davis, R. (1984). Diagnostic reasoning based on structure and function. *Artificial Intelligence*, 24:347–410.
- Eshelman, L. (1988). MOLE: A knowledge-acquisition tool for cover-and-differentiate systems. In Marcus, S., editor, *Automating Knowledge Acquisition for Expert Systems*, pages 37–80. Kluwer, Boston.
- Friedrich, G., Gottlob, G., and Nejdl, W. (1990). Physical impossibility instead of fault models. In *Proc. Eighth National Conf. on Artificial Intelligence*, pages 331–336, Boston.
- Genesereth, M. R. (1984). The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24:411–436.
- Josephson, J. R., Chandrasekaran, B., Smith, J. W., and Tanner, M. C. (1987). A mechanism for forming composite explanatory hypotheses. *IEEE Trans. Systems, Man, and Cybernetics*, 17(3):445–454.
- Levesque, H. J. and Brachman, R. J. (1985). A fundamental tradeoff in knowledge representation and reasoning. In Brachman, R. J. and Levesque, H. J., editors, *Readings in Knowledge Representation*, pages 42–70. Morgan Kaufmann, San Mateo, CA.
- Pearl, J. (1987). Distributed revision of composite beliefs. *Artificial Intelligence*, 33(2):173–215.
- Peng, Y. and Reggia, J. A. (1987). A probabilistic causal model for diagnostic problem solving. *IEEE Trans. on Systems, Man, and Cybernetics*, 17(2):146–162. Part II appears in SMC-17(3):395–406.
- Poole, D. (1989). Explanation and prediction: An architecture for default and abductive reasoning. *Computational Intelligence*, 5(2):97–110.
- Reggia, J. A., Nau, D. S., and Wang, P. (1983). Diagnostic expert systems based on a set covering model. *Int. J. Man-Machine Studies*, 19(5):437–460.
- Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95.
- Selman, B. and Levesque, H. J. (1990). Abductive and default reasoning: A computational core. In *Proc. Eighth National Conf. on Artificial Intelligence*, pages 343–348, Boston.
- Thagard, P. (1989). Explanatory coherence. *Behavioral and Brain Sciences*, 12(3):435–467.

A New Formalisation Of Subsumption In Frame-Based Representation Systems

Jacqueline Castaing
 Lipn/.Csp.
 Université Paris-Nord
 Avenue J-B Clément
 93430 Villetaneuse
 France

Abstract

Computing subsumption is a very common operation in knowledge-based systems; as, it is not desirable to have components of knowledge-based systems that may take an extremely long time. This paper presents a new formal system based on Girad's linear logic, that has a decidable algorithm for determining subsumption. We propose to analyse subsumption with the help of the linear sequent $B \vdash A$. We extract from the proof of this sequent, a criterion to characterize subsumption. We show why even a minor alteration of the expressive power, may turn the subsumption to be undecidable.

1 INTRODUCTION

In this article, we propose a novel approach to the subsumption problem (Brachman and Smolze, 1984), (Brachman and Gilbert, 1985), (Schmidt-Schaub, 1989), which is undoubtedly one of the most important problems in knowledge representation languages. Informally, one concept subsumes another if the first is more general than the other. The subsumption problem has been particularly analysed by Brachman and Levesque (Brachman and Levesque, 1985), (Levesque and Brachman, 1987), and Patel-schneider (Patel-Schneider, 1985, 1988). Two main results can be underlined. First, standard logic is not adequate to formalise the subsumption relation, and we can remark that no syntactical approach proposed so far is satisfactory. Second, a trade-off between expressive power of the representation language and computational tractability of subsumption must be found.

From a fragment of Girard's linear logic (Girard, 1987, 1989), we define decidable linear theories, denoted by FLL, in which the answer to the question "does the concept A subsume the concept B" is given by the proof of the linear sequent $B \vdash A$. Such an approach is made possible by forbidding the **Contraction** and **Weakening** rules in FLL theories. The absence of these rules allows us to make an accurate analysis of the subsumption problem and so, to define a simple criterion of subsumption. As an example, we will show why even a minor alteration of the expressive power (such as introducing in the language a restriction operation which accounts for roles constrained by the types of their fillers), may yield a sudden rise in the complexity of subsumption, and in the general case, may turn the subsumption to be undecidable.

Our paper is presented as follows: in section 2, we introduce the subsumption problem. As our analysis is carried out with the help of sequents, in section 3, we first present the notion of sequent in classical calculus (Gentzen, 1969), (Gallier, 1986). We will particularly stress the role of two structural rules: the contraction and the weakening rules. In sections 4 and 5, after a succinct presentation of linear logic, we will define the FLL theories. Then, we can make clear our analysis of subsumption problem.

2 THE SUBSUMPTION PROBLEM

We introduce the problem with the help of an example borrowed from (Brachman-Levesque, 1984). Let L be the knowledge representation language defined as follows:

(Concept) ::= (Atom) / (And (Concept1) ... (Conceptn))
 / (All (Role) (Concept)) / (Some (Role))
 (Role) ::= (Atom)

Atoms are the names of primitive concepts. And constructions represent conjoined concepts, so for example (And person adult student) represents the concept of something that is at the same time a person, an adult and a student. The All construct provides a value on the fillers of a role: x is in (All $r c$) iff for all y such that $r(x y)$, we have $c(y)$ (a role is a binary predicate, a concept a unary one); thus, (All child doctor) corresponds to the concept of something all of whose children are doctors. The Some operator just guarantees that there will be at least one filler of the role named.

The concept A subsumes the concept B if all instances of the subsumed concept B are also instances of the subsuming concept A. Brachman and Levesque's subsumption algorithm is very simple. Let A and B be two concepts such that: $A = (\text{And } A_1, \dots, A_n)$ and $B = (\text{And } B_1, \dots, B_m)$: if A_i is an atom or a Some, then one of the B_j must be A_i ; if A_i is (All $r c$), then one of the B_j must be (All $r c'$), where c subsumes c' . This algorithm answers A subsumes B in polynomial time.

In order to increase the expressive power of L, the authors introduce in the previous language the new construct Restr: y is (Restr $r c$) of x iff $r(x y)$ and $c(y)$; (Restr child rich) specifies the children who are rich. For example, they consider the following concepts (we give their simplified forms): $A = (\text{And person (All child doctor)})$ and $B = (\text{And person (All child rich) (All (Restr child rich) doctor)})$. They note that computing A subsumes B turns now to be NP-hard.

None of the so far given explanations of this sudden and unexpected "computational cliff" seems to be sufficient for a good understanding of this phenomenon. The numerous analyses proposed are mainly based on two or three, even four-valued semantics (Nebel, 1988), (Patel-Schneider, 16), (Schmidt-Schaub, 17). We propose a syntactical approach of this phenomenon, by means of proofs in FLL theories: we shall point out the important part played by the contraction rule in this sudden rise in computational complexity.

3 CLASSICAL SEQUENT CALCULUS: CONTRACTION AND WEAKENING RULES

The sequent calculus introduced by Gentzen (Gentzen, 1969) is essential to study the properties of proofs. What is gained from linear calculus appears more clearly in a Gentzen-like sequent formalism. For these reasons, we devote this paragraph to introducing the

notion of sequent in classical logic. In the next paragraph, we will give their meaning in linear logic.

Sequents are expressions of the form $B \vdash A$, where $B = (B_1, \dots, B_m)$ and $A = (A_1, \dots, A_n)$ are finite sequences of formulas. The intended meaning of the sequent $B \vdash A$ is B_1 and ... and B_m imply A_1 or...or A_n , where the operators "and", "imply", and "or" have the sense of the classical connectives \wedge , \rightarrow , \vee .

Classical sequent calculus contains:

- i) the identity group which introduces the axioms,
- ii) the cut rule,
- iii) the structural rules composed of the usual exchange rules, the contraction and weakening rules,
- iv) the logical group which gives for each connective, a rule of introduction into the left and into the right side of a sequent.

A proof-tree is a tree whose leaves are axioms, whose nodes are sequent(s) deduced by use of the above rules, and whose root is the sequent to be demonstrated.

Let us introduce the contraction (C) and the weakening (W) rules.

$$\frac{\Gamma, A, A \vdash D}{\Gamma, A \vdash D} \text{ (C-l)} \quad \frac{\Gamma \vdash D, A, A}{\Gamma \vdash D, A} \text{ (C-r)}$$

$$\frac{\Gamma \vdash D}{\Gamma, A \vdash D} \text{ (W-l)} \quad \frac{\Gamma \vdash D}{\Gamma \vdash D, A} \text{ (W-r)}$$

The contraction rule says that every hypothesis (A) can be "used" several times, and leads to a duplication of hypotheses. The weakening rule says that every formula (A) can be added as a hypothesis in a proof and so, is a "useless" hypothesis. Usually, these rules are implicitly applied in standard logic (as structural rules), and no particular attention is paid to them. However, as Girard mentioned in (8), their presence or absence cannot be neglected. For example, in classical logic, we easily admit that the sequents $A \vdash B \vee \neg B$

and $A \wedge \neg A \vdash B$ - which show that every sentence implies a tautology and a contradiction implies everything - are valid. In fact, their proofs are not so obvious, and can be carried out only by means of contraction and weakening (see section-5, example-2). In the next paragraph, we will make clear the role of

these rules is in fact primordial.

In order to make an accurate analysis of the subsumption problem by means of proofs, we look for a system which enables us to carry out 'direct' proofs without duplicating hypotheses or adding some useless ones, with only 'obvious' inferences, those which are linked to the definition of the connectives. Ketonen and Weyhrauch (Ketonen and Weyhrauch, 1984) proved that without contraction, a decidable fragment of predicate calculus can be defined. They only considered the theoretical aspects of the fragment defined. We deal with a practical case, we think (and we will prove it), that linear logic gives an adequate answer to our problem.

4 LINEAR LOGIC: A SUCCINCT PRESENTATION

Linear logic has been seldom applied so far to the formalisation of reasoning systems (Cerrito, 1990), (Masseron and Tolle, 1990). The essential feature of this calculus is the absence of contraction and weakening. An immediate consequence is the distinction of two connectives for conjunction, and two connectives for disjunction.

4.1 LINEAR CONNECTIVES

Let us consider the following (\wedge -r) rules in sequent calculus:

$$\frac{\Gamma \vdash A, D \quad \Gamma' \vdash B, D'}{\Gamma, \Gamma' \vdash A \wedge B, D, D'} \text{ (}\wedge\text{-r1)} \quad \text{or}$$

$$\frac{\Gamma \vdash A, D \quad \Gamma \vdash B, D}{\Gamma \vdash A \wedge B, D} \text{ (}\wedge\text{-r2)}$$

In these rules, contexts Γ , Γ' , and D , D' are dealt with either by juxtaposition or by identification. These rules are equivalent because contraction and weakening hold: from (\wedge -r1), we recover (\wedge -r2).

$$\frac{\Gamma \vdash A, D \quad \Gamma \vdash B, D}{\Gamma, \Gamma \vdash A \wedge B, D, D} \text{ (}\wedge\text{-r1)}$$

(series of contractions on formulas of Γ and D)

$$\Gamma \vdash A \wedge B, D$$

Recovering (\wedge -r1) from (\wedge -r2) is straightforward too by means of the weakening rule.

Without contraction and weakening, the above rules defining the connective \wedge are no longer equivalent, and correspond to two distinct linear connectives, denoted

by \otimes (times) and $\&$ (with). The connective \otimes "inherits" the definition (\wedge -r1), and the connective $\&$ the definition (\wedge -r2) (\otimes requires disjoint contexts whereas $\&$ works with twice the same context). In the same way, the classical connective \vee splits into two different linear connectives, denoted by γ (par) and \oplus (plus). Then, classical implication gives birth to linear implication, denoted by \multimap . Classical negation is replaced by linear negation, denoted by \circ , which satisfies $A^\circ = A$. This connective allows us to translate the linear implication $A \multimap B$ into the formula $A^\circ \gamma B$. All these connectives are linked by de Morgan's relations: $(A \otimes B)^\circ = A^\circ \gamma B^\circ$ and $(A \oplus B)^\circ = A^\circ \& B^\circ$.

The meaning of linear propositions can no longer be given by means of truth tables, as it is the case in classical logic. In fact, it depends not only on truth values of variables, but also on the effort performed in order to carry out proofs (for example, if a hypothesis must be duplicated, the number of its occurrences can be used for evaluating this effort).

4.2 DISTINCTIONS \otimes / $\&$ AND Γ / \oplus

Let us analyse the particular relations between \otimes and $\&$ and γ and \oplus , in the case where either weakening or contraction is accepted. We need to introduce the following rules:

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, (A \otimes B) \vdash \Delta} \text{ (}\otimes\text{-l)}$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash (A \otimes B), \Delta, \Delta'} \text{ (}\otimes\text{-r)}$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, (A \& B) \vdash \Delta} \text{ (&-l1)} \quad \frac{\Gamma, B \vdash \Delta}{\Gamma, (A \& B) \vdash \Delta} \text{ (&-l2)}$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash (A \& B), \Delta} \text{ (&-r)}$$

Technically speaking, weakening says that \otimes is stronger than $\&$. We prove this result (exchange rules can be implicitly used).

$$\frac{\begin{array}{c} A \vdash A \\ A, B \vdash A \end{array} w \quad \begin{array}{c} B \vdash B \\ A, B \vdash B \end{array} w}{\begin{array}{c} A, B \vdash A \& B \\ A \otimes B \vdash A \& B \end{array}} & \& \\ \hline & \otimes & \end{math>$$

Using contraction, we get the reverse implication:

$$\frac{\begin{array}{c} A \vdash A \quad B \vdash B \\ A, B \vdash A \otimes B \end{array}}{\begin{array}{c} A \& B, B \vdash A \otimes B \\ A \& B, A \& B \vdash A \otimes B \end{array}} & \& \\ \hline & c & \end{math>$$

So, with weakening and contraction, the distinction \otimes / $\&$ is no longer justified. We can get a similar conclusion with the connectives par and plus.

If we now consider classical logic, we see that the main meaning of \wedge and \vee as *connectives* is additive: $\wedge = \&$, $\vee = \oplus$; whereas the meaning of \wedge and \vee as *commas* in sequents is multiplicative: $\wedge = \otimes$, $\vee = \gamma$.

5 FLL THEORIES

In the remainder of this article, we consider that the linear fragment used consists of:

- the connective $\&$, which plays the part of the conjunction operation,
- the connective γ , which plays the part of the disjunction operation,
- the linear negation \circ which is restricted to literals.

We will justify our choice after the rules and the examples of proof given below.

5.1 RULES OF INTRODUCTION OF THE CONNECTIVES

Axioms : $A \vdash A$

$$\text{Cut} : \frac{\Gamma \vdash \Delta, A \quad A, \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}$$

Exchange rules:

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, B, A \vdash \Delta} \text{ (Ex-l)} \quad \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, B, A} \text{ (Ex-r)}$$

Logical rules:

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, A^\circ \vdash \Delta} \text{ (\circ-l)} \quad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash A^\circ, \Delta} \text{ (\circ-r)}$$

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma', B \vdash \Delta'}{\Gamma, \Gamma', (A \gamma B) \vdash \Delta, \Delta'} \text{ (\gamma-l)}$$

$$\frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash (A \& B), \Delta} \text{ (\&-r)}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, (A \& B) \vdash \Delta} \text{ (\&-l1)} \quad \frac{\Gamma, B \vdash \Delta}{\Gamma, (A \& B) \vdash \Delta} \text{ (\&-l2)}$$

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash (A \& B), \Delta} \text{ (\&-r)}$$

$$\frac{\Gamma, A(t/x) \vdash \Delta}{\Gamma, \forall x A \vdash \Delta} \text{ (\forall-l)} \quad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash \forall x A, \Delta} \text{ (\forall-r)}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, \exists x A \vdash \Delta} \text{ (\exists-l)} \quad \frac{\Gamma \vdash A(t/x), \Delta}{\Gamma \vdash \exists x A, \Delta} \text{ (\exists-r)}$$

In rules $(\forall-r)$ and $(\exists-l)$, x must not be free in Γ and Δ .

A FLL theory, denoted by \mathcal{T} , is built from the above fragment, by adding a finite set S of sequents (closed under substitution) of the form $B \vdash A$, where B and A are literals (S is the set of proper axioms, and corresponds to the knowledge of a Data Base). We suppose that S has no cycle, i.e, there is no sequence of literals B_0, \dots, B_{n+1} such that : $B_0 = B_{n+1}$, and $B_i \vdash B_{i+1}$, for $i = 0, \dots, n$.

A proof in \mathcal{T} has the same definition as in classical calculus.

Definition: In the cut rule given above, the formula A is the cut-formula. A proof in \mathcal{T} is said to be **cut-free**, if all cut-formulas involved occur in some sequent of S (cut-formulas are literals).

We now give two well-known results applied to FLL theories.

Theorem (Cut elimination): In \mathcal{T} , any provable

sequent has a cut-free proof.

Corollary (Gentzen's subformula property): In a cut-free proof of a sequent $B_1, \dots, B_m \vdash A_1, \dots, A_n$, only subformulas of $B_1, \dots, B_m, A_1, \dots, A_n$, and cut-formulas occurring in some sequent of S , can appear at the nodes of the proof-tree.

5.2 EXAMPLES

We give some examples of proofs, where the set S of proper axioms is empty. We suppose that exchange rules are implicitly used.

Example-1: the sequents $A \& B \vdash A$ and $\forall x A(x) \vdash A(a) \& A(b)$ are provable, and we give their proof-trees:

$$\frac{A \vdash A}{A \& B \vdash A} (\&-l)$$

$$\frac{\begin{array}{c} A(a/x) \vdash A(a) \\ \forall x A(x) \vdash A(a) \end{array} (\forall-l) \quad \begin{array}{c} A(b/x) \vdash A(b) \\ \forall x A(x) \vdash A(b) \end{array} (\forall-l)}{\forall x A(x) \vdash A(a) \& A(b)} (\&-r)$$

Example-2: the sequents $A \vdash B \gamma B^\circ$ et $A \& A^\circ \vdash B$ are not provable: their cut-free proofs should terminate in the following manner:

$$\frac{\begin{array}{c} A, B \vdash B \\ A \vdash B^\circ, B \end{array} (\circ-r)}{A \vdash B^\circ \gamma B} (\gamma-r)$$

$$\frac{A \vdash B}{A \& A^\circ \vdash B} (\&-l) \text{ or } \frac{A^\circ \vdash B}{A \& A^\circ \vdash B} (\&-l2)$$

But, by means of contraction and weakening rules, these sequents turn now to be provable: (formulas involved in these rules are underlined).

$$\frac{\begin{array}{c} B \vdash B \\ A, \underline{B} \vdash B \end{array} (W) \quad \begin{array}{c} A \vdash B^\circ, B \\ A \vdash B^\circ, B \end{array} (\circ-r)}{A \vdash B^\circ \gamma B} (\gamma-l)$$

$$\begin{array}{c} \frac{A \vdash A}{A \vdash A, B} (W) \\ \frac{A \vdash A, B}{A, A^\circ \vdash B} (\circ-l) \\ \frac{A, A^\circ \vdash B}{A, A \& A^\circ \vdash B} (\&-l2) \\ \frac{A, A \& A^\circ \vdash B}{A \& A^\circ, A \& A^\circ \vdash B} (\&-l1) \\ \frac{A \& A^\circ, A \& A^\circ \vdash B}{A \& A^\circ \vdash B} (C) \end{array}$$

Example-3: the sequents $A \otimes B \vdash A$, and $A \otimes B \vdash B$ are not provable.

Example-4: the sequents $A \& (A^\circ \gamma B) \vdash B$ and $A(a) \gamma A(b) \vdash \exists x A(x)$ are not provable.

The examples 1 and 3 make clear the choice of the connective $\&$ as conjunction operation. From $A \otimes B$, we cannot deduce A (or B). The choice of the connective $\&$ for knowledge representation is then essential. The second sequent given in the example-4 shows that the connective γ has the property we need in knowledge representation (see also 15), because we can read the formula $\exists x A(x)$ as: "there exists a known individual for which the predicate A is true".

6 FLL THEORIES: A NEW SUBSUMPTION CRITERION

We now indicate, why Gentzen's subformula property associated to the absence of contraction, yields a decision method in \mathcal{T} .

Let $B_1, \dots, B_m \vdash A_1, \dots, A_n$ be the sequent to be proven. Gentzen's subformula property gives a decision procedure if the length of the sequents involved in a cut-free proof of the sequent $B_1, \dots, B_m \vdash A_1, \dots, A_n$ can be bound. It is the case if the hypotheses are not duplicated. So, the absence of contraction is crucial (see also Ketonen and Weyhrauch, 1984). This guarantees that every formula can be 'used' at most once, and the search procedure for a proof always terminates, either with a success or with a failure. The reader can remark that the weakening rule has no influence upon the decidability problem. In fact, FLL theories turn into full first-order theories if the contraction and weakening rules hold. We now give a decision procedure for a certain class of formulas in \mathcal{T} .

Definition: Every formula which is a disjunction of

literals is a linear clause.

Definition: Let $B_C = \forall z B_1(z) \& \dots \& B_m(z)$ and $A_C = \exists x A_1(x) \& \dots \& A_n(x)$ be two first-order formulas in conjunctive normal form, where z and x can be vectors of variables, $A_i(x)$ and $B_j(z)$ are linear clauses, $A_i(x) = A_{i1}(x) \gamma \dots \gamma A_{ip}(x)$, $B_j(z) = B_{j1}(z) \gamma \dots \gamma B_{jq}(z)$. We say that the sequent $B_C \vdash A_C$ is in **normal form**.

Our decision procedure only deals with sequents in normal forms $B_C \vdash A_C$. We propose to interpret the concepts B and A by means of first-order formulas, as in Brachman-Levesque's interpretation., then, to replace all classical connectives with linear ones. So, sequents in normal form can be obtained from concepts, with the help of a quasi-linear algorithm (see also Patel-Schneider, 1985).

In section II, we have given an informal definition of subsumption. The properties of FLL theories , presented in the section above, allow us to give this formal and general definition of subsumption:

Definition: A subsumes B iff there exists a cut-free proof in \mathcal{T} of the sequent $B \vdash A_C$.

Definition: Let P and Q be two literals. P is said to be **finitely chained to Q** iff the sequent $P \vdash Q$ is provable in \mathcal{T} .

Example: Let $A \vdash B^\circ$ $C \vdash B$ be two proper axioms. Then A is finitely chained to C° .

Definition: Let $P = (P_1 \gamma (P_2 \gamma (\dots (P_{p-1} \gamma P_p) \dots))$ be a linear clause. The number k of literals of P is the length of P .

Lemma: Let $P = (P_1 \gamma (P_2 \gamma (\dots (P_{p-1} \gamma P_p) \dots))$ and $Q = (Q_1 \gamma (Q_2 \gamma (\dots (Q_{q-1} \gamma Q_q) \dots))$ be two linear clauses. There exists a cut-free proof of the sequent $P \vdash Q$ in FLL iff:

i) P and Q have the same length ($p=q$)

ii) there exists a bijection s of the set of indices $(1, p)$ into itself, such that for every $i : 1 \leq i \leq p$, the literal P_i is finitely chained to the literal $Q_s(i)$.

In order to prove this lemma, we propose first to prove the lemma-1, which is slightly different, and then, to conclude by means of the lemma-2.

Lemma-1: Let P be a linear clause of length p , $(P_1$

$\gamma (P_2 \gamma (\dots (P_{p-1} \gamma P_p) \dots))$ and $Q = (Q_1, \dots, Q_q)$ be a sequence of literals. There exists a cut-free proof of the sequent $P \vdash Q_1, \dots, Q_q$ in \mathcal{T} iff:

i) the length of P is equal to the number of literals in Q ($p=q$)

ii) there exists a bijection s of the set of indices $(1, p)$ into itself, such that for every $i : 1 \leq i \leq p$, the literal P_i is finitely chained to the literal $Q_s(i)$.

Lemma-2: Let $P = (P_1 \gamma (P_2 \gamma (\dots (P_{p-1} \gamma P_p) \dots))$ and $Q = (Q_1 \gamma (Q_2 \gamma (\dots (Q_{q-1} \gamma Q_q) \dots))$ be two linear clauses. The sequent $P \vdash Q$ is provable in \mathcal{T} iff the sequent $P \vdash Q_1, \dots, Q_q$ is provable in \mathcal{T} .

The proof of the lemma-2 is obvious, we only present the proof of the lemma-1.

Proof of lemma-1:

a) Necessary condition: Let us suppose that there exists a cut-free proof of the sequent

$(P_1 \gamma (P_2 \gamma (\dots \gamma (P_{p-1} \gamma P_p) \dots)) \vdash Q_1, \dots, Q_q$. Let us then show that conditions i) and ii) are satisfied.

We essentially use the induction principle applied to proof-trees.

Basic case: the proof-tree is a one-node tree. The sequent $P_1 \vdash Q_1$ is an axiom of \mathcal{T} , then conditions i) and ii) are satisfied.

General case: let T be the proof-tree of the sequent $(P_1 \gamma (P_2 \gamma (\dots \gamma (P_{p-1} \gamma P_p) \dots)) \vdash Q_1, \dots, Q_q$. The last rule applied is either an exchange rule or the rule $(Y-I)$. The exchange rules can only permute the literals of the sequence Q_1, \dots, Q_q . We can always consider that the last rule applied is $(Y-I)$. The proof-tree T is then the following one:

T1

$P_1 \vdash Q_1 \quad (P_2 \gamma (\dots \gamma (P_{p-1} \gamma P_p))) \vdash Q_2, Q_j, Q_q$

$(P_1 \gamma (P_2 \gamma (\dots \gamma (P_{p-1} \gamma P_p)))) \vdash Q_1, \dots, Q_q$

T 2

T2 is the proof-tree of the sequent $(P_2 \gamma (\dots \gamma (P_{p-1} \gamma P_p))) \vdash Q_2, \dots, Q_j, \dots, Q_q$. By the induction hypothesis, we have i) $p-1 = q-1$, then $p = q$; ii) there exists a bijection s of the set of indices $(2, p)$ into itself such that for $i : 2 \leq i \leq p$, the literal P_i is finitely

chained to the literal $Q_{S(i)}$. T_1 is the proof-tree of the sequent $P_1 \vdash Q_1$. We can consider that: i) $s(1) = 1$; ii) P_1 is finitely chained to Q_1 (see the definition).

b) Sufficient condition: let us suppose that the two conditions i) and ii) are satisfied. We show that the sequent $(P_1 \gamma (P_2 \gamma (\dots \gamma (P_{p-1} \gamma P_p) \dots)) \vdash Q_1, \dots, Q_p$ is provable. The proof consists in

- iterating the application of the rule (γ -I):

$$\begin{array}{c} P_{p-1} \vdash Q_{S(p-1)} \quad P_p \vdash Q_{S(p)} \quad (\gamma\text{-I}) \\ (P_{p-1} \gamma P_p) \vdash Q_{S(p-1)}, Q_{S(p)} \\ \hline \hline \end{array}$$

$$(P_1 \gamma (P_2 \gamma (\dots \gamma (P_{p-1} \gamma P_p) \dots)) \vdash Q_{S(1)}, \dots, Q_{S(p)})$$

to prove the sequent

$$(P_1 \gamma (P_2 \gamma (\dots \gamma (P_{p-1} \gamma P_p) \dots)) \vdash Q_{S(1)}, \dots, Q_{S(p)}.$$

- then, by means of the exchange rules to obtain the proof-tree of the sequent

$$(P_1 \gamma (P_2 \gamma (\dots (P_{p-1} \gamma P_p) \dots)) \vdash Q_1, \dots, Q_p.$$

End of Proof

Definition: Let P and Q be two linear clauses. We say that P is **chained** to Q , iff there exists a cut-free proof in \mathfrak{T} of the sequent $P \vdash Q$.

Example: Let $P = P_1 \gamma P_2$, $Q = Q_1 \gamma Q_2$ be two formulas, and $P_1 \vdash R$, $R \vdash Q_1$, $P_2 \vdash Q_2$ be proper axioms of \mathfrak{T} . P is chained to Q because P_1 is finitely chained to Q_1 through the literal R , and P_2 is finitely chained to Q_2 as a proper axiom of the system.

Theorem (subsumption criterion): A subsumes B iff $Ac = \exists x A_1(x) \& \dots \& A_n(x)$, and $Bc = \forall z B_1(z) \& \dots \& B_m(z)$ satisfy the following condition (C): there exists α , a substitution for x such that for each A_i , $1 \leq i \leq n$, there exists some B_j , $1 \leq j \leq m$, and β , a substitution for z , such that $B_j \beta$ is chained to $A_i \alpha$.

We just sketch the proof of the necessary condition: A subsumes $B \Rightarrow (C)$.

We use the induction principle applied to proof-trees. In case of a one-node tree, the claim is obvious. In the induction step we must consider numerous cases. We restrict ourselves to the case where the last rule applied

is a $\&$ -r rule. Let T be the following proof-tree of the sequent $Bc \vdash Ac$:

$$\begin{array}{cc} T_1 & T_2 \\ \hline \underline{B_1 \& \dots \& B_m \vdash A_1 \ B_1 \& \dots \& B_m \vdash (A_2 \& \dots \& A_n)} & (\&-r) \\ B_1 \& \dots \& B_m \vdash A_1 \& (A_2 \& \dots \& A_n) \end{array}$$

The trees T_1 and T_2 are the proof-trees of the sequents $B_1 \& \dots \& B_m \vdash A_1$ and $B_1 \& \dots \& B_m \vdash A_2 \& \dots \& A_n$. By the induction hypothesis, there exists α , a substitution for x such that: for A_1 , there exists some B_j , $1 \leq j \leq m$, and a substitution β , such that $B_j \beta$ is chained to $A_1 \alpha$; for each A_i , $2 \leq i \leq n$, there exists some B_j , $1 \leq j \leq m$, and a substitution β such that $B_j \beta$ is chained to $A_i \alpha$. So, condition (C) holds.

Implementation

A matching algorithm can be easily derived from condition (C). Let p be the greatest length of the A_i , for $i : 1 \leq i \leq n$, and r be the cardinality of the set S of proper axioms. Then, our algorithm computes subsumption in polynomial time proportional to the arguments m, n, r, p .

7 APPLICATION

We first indicate how we put the concepts into conjunctive normal form in a FLL theory. We limit ourselves to the knowledge representation language given in section 2.

7.1 CONCEPTS IN \mathfrak{T}

(And Concept₁, ..., Concept_n) is a conjunction of concepts: its prenex conjunctive normal form in a FLL theory is $\forall x (C_1(x) \& \dots \& C_n(x))$, where C_i , for $i : 1 \leq i \leq n$, is the linear clause corresponding to Concept_i.

(All Role Concept) is considered in Brachman-Levesque's interpretation as the first-order formula $\forall xy (R(x, y) \rightarrow C(y))$. After replacing the classical connective \rightarrow with linear implication, we obtain the formula $\forall xy (R(x, y) \circ C(y))$, which has the prenex normal form: $\forall xy (R^\circ(x, y) \gamma C(y))$.

(Some Role) is interpreted as the following formula: $\forall x \exists y R(x, y)$.

(Restr Role Concept) is first put into the first-order formula $\forall xy (R(x, y) \wedge C(y))$, and the corresponding formula in a FLL theory is $\forall x y (R(x, y) \wedge C(y))$.

Let $A = (\text{And } CA_1 \dots CA_n)$ and $B = (\text{And } CB_1 \dots CB_m)$ be two concepts. Let A_i , for $i: 1 \leq i \leq n$, and B_j , for $j: 1 \leq j \leq m$, be the linear clauses corresponding to CA_i and CB_j (obtained as indicated above). In this particular case, the relation "A subsumes B" is analysed by means of the sequent in normal form $\forall z B_1 \wedge \dots \wedge B_m \vdash \exists x A_1 \wedge \dots \wedge A_n$ obtained in the following way: from the formula $\forall u (\forall z B_1 \wedge \dots \wedge B_m \rightarrow \exists x A_1 \wedge \dots \wedge A_n)$ which expresses that all instances of the concept B are also instances of the concept A, we define the sequent $\vdash \forall u (\forall z B_1 \wedge \dots \wedge B_m \rightarrow \exists x A_1 \wedge \dots \wedge A_n)$. then, by applying the (\forall -r) rule, we deduce the sequent $\vdash (\forall z B_1 \wedge \dots \wedge B_m \rightarrow \exists x A_1 \wedge \dots \wedge A_n)$, which yields the sequent $\forall z B_1 \wedge \dots \wedge B_m \vdash \exists x A_1 \wedge \dots \wedge A_n$.

7.2 EXAMPLES

In our first example, we again consider the Brachman-Levesque's algorithm (see section 2), and we propose to show that in the particular conditions of this algorithm, our decision procedure can give a direct answer too.

Example-1: Let A and B be two concepts, $A = (\text{And } CA_1 \dots CA_n)$, $B = (\text{And } CB_1 \dots CB_m)$. If CA_i , for $i: 1 \leq i \leq n$, is an atom or a Some, and there exists CB_j , for $j: 1 \leq j \leq m$, such that CB_j is CA_i , then the linear clause B_j is obviously chained to A_i . If CA_i is the concept (All Role Concept $_i$) and one of the CB_j is (All Role Concept $_j$) such that Concept $_i$ subsumes Concept $_j$, it is obvious that the linear clause $(R^o(x, y) \wedge C_j(y))$ (corresponding to CB_j) is chained to the linear clause $(R^o(x, y) \wedge C_i(y))$ (corresponding to CA_i), provided that the sequent $C_j(y) \vdash C_i(y)$ is provable, which is the case because Concept $_i$ subsumes Concept $_j$.

Example-2: We now show how our algorithm can deal with some hierarchy on roles.

Let $A = (\text{And Person (All Cousin Rich)})$ and $B = (\text{And Person (All Friend Doctor)})$ be two concepts. A can be rendered as a noun phrase like "person each of whose cousin is rich", and B like "person each of whose

friend is a doctor".

The relation "A subsumes B" is analysed by means of the following sequent $B_C \vdash A_C$ where :

$$B_C = \forall u (\text{Person}(x) \wedge (\text{Friend}^o(x, u) \wedge \text{Doctor}(u))).$$

$$A_C = (\text{Person}(x) \wedge (\text{Cousin}^o(x, y) \wedge \text{Rich}(y))).$$

Let us suppose that the set of known facts is the following :

- Rich subsumes Doctor, which is interpreted in \mathcal{T} by means of the proper axiom $\text{Doctor}(y) \vdash \text{Rich}(y)$,
- the role Cousin is linked to the role Friend by means of a hierarchical relationship, which can be interpreted in \mathcal{T} with the help of the following proper axiom: $\text{Cousin}(x, y) \vdash \text{Friend}(x, y)$.

The sequent

$$\forall u (\text{Person}(x) \wedge (\text{Friend}^o(x, u) \wedge \text{Doctor}(u))) \vdash (\text{Person}(x) \wedge (\text{Cousin}^o(x, y) \wedge \text{Rich}(y)))$$

is then provable because:

- a) $\text{Person}(x)$ is chained to $\text{Person}(x)$

$(\text{Person}(x) \vdash \text{Person}(x))$ is an axiom)

- b) $(\text{Friend}^o(x, u) \wedge \text{Doctor}(u))$ with the substitution y/u is chained to $(\text{Cousin}^o(x, y) \wedge \text{Rich}(y))$, (by hypothesis the literal $\text{Cousin}(x, y)$ is finitely chained to the literal $\text{Friend}(x, y)$, and the literal $\text{Doctor}(y)$ is finitely chained to literal $\text{Rich}(y)$ too).

So, A subsumes B, and it means that everybody whose friends are doctors, have rich cousins, provided that doctors are rich, and to be a cousin means to be a friend too!.

8 CONTRACTION AND COMPUTATIONAL COMPLEXITY

From the results obtained in the previous section, we can now underline the link between the presence of the contraction rule and the rise in complexity of subsumption.

8.1 CONTRACTION / RISE IN COMPLEXITY

The sudden and unexpected "computational cliff" of subsumption underlined by Brachman and Levesque (see section 2) can now be explained by analysing the complexity of the search procedure for a proof of the sequent $B_C \vdash A_C$ in \mathcal{T} . We consider the following cases:

i) B_C and A_C satisfy condition (C) (the contraction rule is absent): the sequent $B_C \vdash A_C$ is provable in polynomial times, then the complexity of subsumption is polynomial.

ii) Bc and Ac do not satisfy condition (C): let us suppose that the sequent $B_c \vdash A_c$ is provable (for example, by means of an approach based on semantics), and the proof of the sequent $B_c \vdash A_c$ necessitates the use of the contraction rule, (and possibly of the weakening rule): the search procedure for a proof can make sequents of the form $\forall z B(z,a) \vdash \Delta$, (or of the form $\Gamma \vdash \exists x A(x,a)$) appear at the nodes of the search-tree. Let us consider the case, where the sequent $\forall z B(z,a) \vdash \Delta$ appears at a node of the search-tree: the search procedure can go back-up the tree by applying the universal and contraction rules. We can be left with the following tree:

$$\frac{\underline{B(b/z, a), \forall z B(z,a) \vdash \Delta} \quad (\forall\text{-I})}{\underline{\forall z B(z,a), \forall z B(z,a) \vdash \Delta} \quad (\text{C-I})}$$

The use of contraction may open a branch which terminates with a failure. Some back-tracking is then necessary. The complexity of the subsumption in this case is NP-hard.

iii) the sequent $B_C \vdash A_C$ is not provable, then, the use of the contraction rule may lead to duplicate infinitely the same formulas in the case where the set of instantiation terms (such as b) is infinite (for example in presence of functions) :

1

$$\frac{\frac{\frac{B(b/z, a), \forall z B(z, a), \forall z B(z, a) \vdash \Delta}{B(b/z, a), \forall z B(z, a) \vdash \Delta} (C-I)}{\forall z B(z, a), \forall z B(z, a) \vdash \Delta} (\forall I)}{\forall z B(z, a) \vdash \Delta} (C-I)$$

The subsumption turns to be undecidable (first-order theories are undecidable (Cook, 1971)).

8.2 EXAMPLE

Let us come back now to Brachman-Levesque's example (see section 2):

A = (And person (All child doctor)) and B = (And person (All child rich) (All (Restr child rich) doctor)). Brachman-Levesque showed that A subsumes B by using an approach based on two-valued semantics. In fact, their proof uses the contraction rule. We point it out.

$$\begin{aligned} Bc &= \forall u (B1 \wedge B2 \wedge B3) = \forall u (\text{person}(x) \wedge (\text{child}^o(x \\ u) \wedge \text{rich}(u)) \wedge (\text{child}^o(x \ u) \wedge \text{rich}^o(u) \wedge \text{doctor}(u))). \\ Ac &= (A1 \wedge A2) = (\text{person}(x) \wedge (\text{child}^o(x \ y) \wedge \\ \text{doctor}(y))); \end{aligned}$$

- B1 is chained to A1, but neither B2 nor B3 is chained to A2, then, condition (C) does not hold. But, we can "force" the proof of the sequent $B_C \vdash A_C$ by means of the contraction rule. We give the proof-tree (we do not mention variables in the formulas for simplifying the notations):

	B2, B3 \vdash A2 ----- (& -12)
	B2, B2&B3 \vdash A2 ----- (&-11)
axiom	B2&B3, B2&B3 \vdash A2 ----- (C)
B1 \vdash A1 ----- (&-I1)	(B2&B3) \vdash A2 ----- (&-I2)
(B1&(B2&B3)) \vdash A1 ----- (\forall -I)	(B1&(B2&B3)) \vdash A2 ----- (\forall -I)
$\forall_u (B1 \& (B2 \& B3)) \vdash A1$	$\forall_u (B1 \& (B2 \& B3)) \vdash A2$
	----- (&-r)
$\forall_u (B1 \& (B2 \& B3)) \vdash (A1 \& A2)$	

Let us focus on the proof of the sequent $B2, B3 \vdash A2$. We abbreviate $\text{child}(x,y)$ by C , $\text{rich}(y)$ by R , and $\text{doctor}(y)$ by D .

$$\begin{array}{c}
 \frac{\text{R}^\circ \vdash \text{R}^\circ \quad \text{D} \vdash \text{D}}{\text{-----} \quad (\gamma\text{-l})} \\
 \frac{\text{R}^\circ \gamma \quad \text{D} \vdash \text{R}^\circ, \text{D}}{\text{-----} \quad (\circ\text{-l})} \\
 \frac{\text{C}^\circ \vdash \text{C}^\circ \quad \text{R}, (\text{R}^\circ \gamma \text{D}) \vdash \text{D}}{\text{-----} \quad (\gamma\text{-l})} \\
 \frac{\text{C}^\circ \vdash \text{C}^\circ \quad \text{R}, \text{C}^\circ \gamma (\text{R}^\circ \gamma \text{D}) \vdash \text{C}^\circ, \text{D}}{\text{-----} \quad (\gamma\text{-l})} \\
 \frac{\text{C}^\circ \gamma \quad \text{R}, \text{C}^\circ \gamma (\text{R}^\circ \gamma \text{D}) \vdash \text{C}^\circ, \text{C}^\circ, \text{D}}{\text{-----} \quad (\text{C})} \\
 \frac{\text{C}^\circ \gamma \quad \text{R}, \text{C}^\circ \gamma (\text{R}^\circ \gamma \text{D}) \vdash \text{C}^\circ, \text{D}}{\text{-----} \quad (\gamma\text{-r})} \\
 \frac{\text{C}^\circ \gamma \quad \text{R}, \text{C}^\circ \gamma (\text{R}^\circ \gamma \text{D}) \vdash \text{C}^\circ \gamma \text{D}}{\text{-----}}
 \end{array}$$

In this particular example, the complexity of subsumption is NP-hard.

9 CONCLUSION

Our formal system can be used to analyse subsumption in KL-ONE (Brachman and Smolze, 1985), KRYPTON (Brachman and Gilbert, 1985), NIKL (Kazmareck and Bates, 1986), BACK (Nebel, 1988), or in Patel-Schneider's TERMINOLOGICAL'S LOGICS (Patel-Schneider, 1985, 1988). We are now investigating a new subsumption criterion (which is less drastic than the one we proposed in this paper, for example by allowing the weakening rule), and a new subsumption language which fits the criterion proposed.

Acknowledgements

We wish to thank Jacqueline Vauzeilles for helpful comments.

References

- 1 Brachman R.J, and Levesque H.J. (1984) : "The Tractability of Subsumption in Frame-Based Description language" Proceedings AAAI-84, August 84, pp34-37.
- 2 Brachman R.J, and Smolze J.G. (1985) : "An overview of the KL-ONE Knowledge Representation System. Cognitive Sci. 9(2) (1985) 171- 216.
- 3 Brachman R.J, Gilbert V.P, and Levesque H.J. (1985) : "An Essential Hybrid Reasoning System:Knowledge and Symbol Level Accounts of KRYPTON" Proc. 9th IJCAI (1985) Los Angeles. pp 532-539.

4 Cerrito S. (1990): " A linear semantics for Allowed Logic Programs"

Proc. 5th Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, 1990, 219-227.

5 Cook S. A. (1971) : "On the complexity of Theorem-proving Procedures"

Proc. 3rd Annual ACM Symposium On Theory Computing 1971 pp 151-158.

6 Gallier J. (1986) : "Logic for Computer Science: Foundations of Automatic Theorem Proving, Harper and Row, New York (1986).

7 Girard J. Y. (1987) : "Linear Logic" Theoretical Computer Science 50 (1987) .pp1-102.

8 Girard J. Y. (1989) :" Towards a Geometry of Interaction"

Proc. AMS Conference on Categories, Logic and Computer, Contemporary Mathematics 92, AMS 1989.

9 Gentzen G. (1969) :" The Collected Papers of Gerhard Gentzen"

Ed. E; Szabo, North-Holland, amsterdam (1969).

10 Kazmareck T.S, Bates R, and Robbins G. (1986): "Recent Developments in NIKL"

Proc. AAAI-86. Philadelphia, 978-985.

11 Ketonen J., and Weyhrauch R. (1984) : "A decidable fragment of predicate calculus"

Theoretical Computer Science 32:3, 1984.

12 Levesque H.J, and Brachman R.J. (1987): "Expressiveness and Tractability in Knowledge Representation and Reasoning, Comp. Intell. 3 (2) (1987) pp78-93.

13 Masseron M., Tollu C., and Vauzeilles J. (1990) : "Generating plans in linear logic"

Proc. FST & TCS 10, Bengalore (India), Dec. 1990.

14 Nebel B. (1988) : "Computational Complexity of Terminological Reasoning in BACK".

Artificial Intelligence 34 (1988) pp371-383.

15 Patel- Schneider P.F. (1985) : "A Decidable First-Order Logic for Knowledge Representation"

Proceedings 9th. IJCAI (1985). Los Angeles. pp 455-458.

16 Patel- Schneider P.F: "A Four-Valued Semantics for Terminological Logics"
Artificial Intelligence. 36 (1988) pp 319- 353.

17 Schmidt-Schaub M.: "Subsumption in KL-ONE is Undecidable"
First International Conference on Principles of Knowledge Representation.1989. pp 421-431.

Integrating Efficient Model-Learning and Problem-Solving Algorithms in Permutation Environments

Prasad Chalasani
 School of Computer Science
 Carnegie Mellon University
 Pittsburgh, PA 15213
 chal+@cs.cmu.edu

Oren Etzioni*
 Department of Computer Science
 and Engineering, FR-35
 University of Washington
 Seattle, WA 98195
 etzioni@cs.washington.edu

John Mount
 School of Computer Science
 Carnegie Mellon University
 Pittsburgh, PA 15213
 jmount+@cs.cmu.edu

Abstract

Model-learning and problem-solving algorithms can only be integrated when the output of the former fulfills the expectations of the latter. Korf's Macro-Table (KMT) algorithm requires serial or total decomposability of the operators in its world model. The problem of detecting decomposability has been open since Korf introduced the KMT algorithm in 1983. We present a polynomial-time algorithm for detecting total decomposability, and an exponential-time algorithm for detecting serial decomposability in update graphs, an efficiently-learnable model of external environments. A number of additional results are presented including a transformation of update-graph models to permutation group problems that can be handled by Furst, Hopcroft and Luks's (FHL) permutation-group algorithm. The results facilitate the integration of Schapire and Rivest's update-graph-learning algorithm with the KMT and FHL algorithms.

1 INTRODUCTION

An autonomous agent seeking to solve problems in an external environment faces two related challenges: inferring an internal model of the environment and using the model to efficiently solve problems in the environment. Whereas previous work has addressed these challenges in isolation ([RS87], [Sch88], [Kor85], [FHL80]), this paper shows how to tackle them in concert.

In [RS87], Rivest and Schapire present a polynomial-time algorithm that reliably learns an internal model for a class of finite automaton environments, called permutation environments. They show that their internal model, called the *update graph*, is a natural encoding of the behavior of finite automaton environments, and could be as small as logarithmic in the size of the state graph of the automaton. However, they do not consider the possibility of using the update graph as a state representation for solving problems.

Two candidates for use in problem solving using update graphs are: Korf's [Kor85] Macro-Table (KMT) algorithm, and Furst, Hopcroft, and Luks's (FHL) [FHL80] permutation-group algorithm. However, an update graph representation may not necessarily satisfy the assumptions these algorithms make about their input representation. The KMT algorithm can only be applied to a problem representation that has serially/totally decomposable actions. The FHL algorithm only solves permutation group problems. So an autonomous agent that uses these problem-solving algorithms must be able to efficiently perform two tasks: (a) *detect* whether the learned update graph satisfies the input requirements of the algorithms, or (b) *Transform* the update graph into a representation which does satisfy those properties.

Our main aim in this paper is to show polynomial-time algorithms that can be used to accomplish these tasks under different conditions.

*This paper describes research done primarily at Carnegie Mellon University's School of Computer Science. The second author was supported by an AT&T Bell Labs Ph.D. Scholarship.

This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U.S. Air Force, Wright-Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government.

2 OVERVIEW

An update graph is a graph model of a finite automaton environment, whose node-values (called the *labeling*) uniquely characterize the state of a finite automaton, so an update graph can be used as a state representation for problem-solving. A problem to be solved using an update graph can be posed as a goal-labeling. In a permutation environment, executing action b has the effect of permuting the node-values according to a permutation π_b . Thus, problem-solving with the update graph of a permutation environment can be viewed abstractly as the following problem, which we call the *general permutation problem*, GPP: *Given a set of permutations Π of degree n , two n -dimensional state vectors V_o (initial vector) and V_g (goal vector), efficiently find a short sequence of permutations from Π that transform V_o into V_g .*

Two algorithms exist that solve variants of the GPP: (a) Furst, Hopcroft, Luks's [FHL80] (FHL) algorithm solves a restricted version of GPP in which the values of the components of V_o are distinct, and (b) Korf's Macro Table (KMT) algorithm solves problems in which actions (not necessarily permutations) are serially/totally decomposable.

Our main results are polynomial-time algorithms that analyze the structure of a GPP, in order to:

- (a) *detect* reliably whether the permutations Π are totally decomposable with respect to V_o , (which implies they are serially decomposable) so that the KMT algorithm can be applied,
- (b) *reduce* the GPP to a new GPP of smaller size that may satisfy the requirements of the FHL or KMT algorithms, and
- (c) *transform* the GPP to another GPP so that either the FHL or the KMT algorithms can be applied.

3 DEFINITIONS

The “world” in which an intelligent agent finds itself can have different degrees of structure. The language of group theory provides a succinct way of describing regularities in the environment.

A **group** [Mac68] is defined by a set G and a binary operation \circ such that (a) G is closed under \circ , (b) the operation \circ is associative, (c) every element of G has a inverse, and (d) there exists an identity element e in G . If there is a subset X of G such that all elements of G can be obtained by repeatedly applying the \circ operator to elements of X , then X is said to generate G , and the elements of X are called *generators* of G .

By a **permutation** π of an arbitrary set X we shall mean a bijection from X to itself. Any given collection of permutations over the set X generates a permutation group under the operation of composition of permutations, and the permutations in the group are

said to have *degree* $n = |X|$. We use the notation $\langle \Pi \rangle$ to denote the group generated by the permutations Π . Note that the collection of *all* permutations of X forms a group S_X under composition of functions. In particular, if $X = \{1, 2, \dots, n\}$, we denote the group of all permutations over X by S_n , and call S_n the **symmetric group** of degree n . If g is a permutation of degree n , then the result of “applying the permutation g to an n -dimensional vector V ” is a new vector $V' = g(V)$ where $g(i) = j$ implies $V'[j] = V[i]$, for $i, j = 1, 2, \dots, n$.

A **finite automaton environment** (FAE) \mathcal{E} is characterized by a tuple $(Q, B, P, \delta, \gamma)$, where Q is a finite set of states, P is a finite set of predicates, B is a finite set of basic actions, $\delta : Q \times B \rightarrow Q$ is the state transition function, $\gamma : Q \times P \rightarrow \Sigma$ is the sensor function that associates predicate-values with states, Σ being the (finite) set of possible predicate values. We use the notation qa to denote the (unique) final state reached after applying an action-sequence $a \in B^*$ from state q . Note that in a finite automaton environment, for all states q , and for all $b \in B$, qb must be well defined, and unique.

An environment with more structure is the **Permutation Environment**(PE), [RS87], which is a finite automaton environment where the transition function of each action b : $\delta(., b) : Q \rightarrow Q$ is a bijection on Q . This definition implies that for each state of the automaton, there is exactly one incoming and one outgoing b -arrow, for every action $b \in B$. Thus for a PE, all the actions B are *invertible*.

Rivest and Schapire [RS87] devise an efficient inference algorithm for permutation environments.

Examples:

An example of an FAE is the 8-puzzle environment, \mathcal{E}_8 , whose action-set $\{L, R, U, D\}$ corresponds to the four directions (left, right, up, down) in which the blank can be moved. The predicates are p_i , $i = 1, 2, \dots, 8$ whose value is the position of tile i . The actions of \mathcal{E}_8 are not very regular. For example, the action L has “different kinds of effects” in different states. When the blank is in the leftmost position, this operator has no effect. (In order to model the 8-puzzle as a finite automaton, we *must* define a next state for every action, so when the action is “not applicable” in the usual sense, we say that the next-state is the current state itself.) The 8-puzzle environment \mathcal{E}_8 is clearly not a PE because, for example, two different states can lead to the same state through the action L : A state s whose bottom row is $[B \ 1 \ 2]$, and a state which is the same as s except that its bottom row is $[1 \ B \ 2]$. (B represents the blank position). So, it can be pointed out that while the 8-puzzle operators are “invertible”, they are only invertible in the colloquial sense. They are not invertible in the corresponding finite automaton.

The Rubik’s Cube environment \mathcal{E}_{Rubik} with the usual operators, and predicates with values that each rep-

represent the position of a given square on the cube is a PE. This is because the effects of the operators are uniform: any given state is reachable from exactly one other state, through a given operator. The Rubik's Cube environment $\mathcal{E}_{\text{Rub}2}$ with the same operators, and predicates whose values each represent which square is in a fixed position is also a PE by the same reasoning. ■

We now turn to another dimension of structure and regularity in an environment.

3.1 SERIAL & TOTAL DECOMPOSABILITY

Consider a *state vector representation* for a FAE where each state is associated with an assignment of values to n state variables. A given assignment to the n variables is called a *state vector*. Note that different states need *not* have distinct state vectors. Now, suppose the n variables are ordered v_1, v_2, \dots, v_n . When a basic action $b \in B$ is applied to the automaton, the variables acquire new (or possibly the same) values. Let c_i denote the new value acquired by v_i . Then the action b is said to be *serially decomposable* w.r.t. the ordering of those variables if each c_i is expressible as a function only of the variables v_j such that $j \leq i$. If each c_i is expressible as a function of only v_i , then the action b is said to be *totally decomposable*.

Examples: The variables from the environment $\mathcal{E}_{\text{Rub}1}$ are totally decomposable because the next position of a face on the cube is a function of its current position and the operator applied. However the variables from the environment $\mathcal{E}_{\text{Rub}2}$ are not totally decomposable because on the Rubik's cube the next face to occupy a position is not a function of the face currently in the position and the operator applied. It is not hard to see the variables of $\mathcal{E}_{\text{Rub}2}$ are not serially decomposable for any ordering. ■

3.2 UPDATE GRAPHS

In [RS87], Rivest and Schapire show a probabilistic polynomial-time algorithm to infer an internal model of a permutation environment, given the ability to apply actions to the environment and sense predicates from it. Their internal model is called the update graph. We describe briefly the structure of this model.

An update graph can be defined for any FAE $\mathcal{E} = (Q, B, P, \delta, \gamma)$. Every node of an update graph has a value in a given state, and the combination of node values of an update graph uniquely characterizes the state of the automaton. For each predicate p_i there is a node v_i that gives the value of the predicate in any state. Also, for every node and every action $b \in B$, there is exactly one directed edge labeled b going into that node. In the case of a permutation environment(PE), there is exactly one outgoing edge labeled b from any

node. When an action b is applied to the automaton, the new values of the nodes are given by simply moving the node values in the direction of the b -edges: the new value of a node v_i is simply the current value of the unique node v_j which has a b -edge going out from it to v_i . Thus, given the initial labeling (node-values) of the update graph, it is possible to determine the labeling (and hence the values of the predicates P) after any sequence of actions $a \in B^*$. In the case of a PE, the update graph is completely characterized by:

(a) an n -dimensional vector V_o that gives the initial labeling of the graph, n being the number of nodes of the update graph.

(b) a set Π of permutations of degree n that correspond to the actions.

We will henceforth denote an update graph by the pair (Π, V_o) . Thus, for a PE, the update graph provides a state vector representation which has a particularly simple behavior: the values of the state variables are rearranged when actions are performed. Our algorithms take advantage of this simple behavior.

4 THE GENERAL PERMUTATION PROBLEM (GPP)

We can now state the general problem that we are concerned with in this paper. Once the agent infers an update graph for the permutation environment, it is presented with problems of the following form: "Achieve a particular set of values of the predicates P ." This is equivalent to specifying goal values on a subset of the nodes of the update graph. We consider, the probably more natural, version of this problem where the goal specifies the values of *all* nodes of the update graph. Thus the problem is, "given an update graph (Π, V_o) , achieve a labeling V_g , using a sequence of permutations (actions) from Π ".

If $G = \langle \Pi \rangle$ is the group generated by Π , then we use the notation $G(V_o)$ to denote the set of possible n -dimensional vectors that are reachable from V_o by permutations of Π .

Any problem on an update graph of a PE can be viewed abstractly as the following *General Permutation Problem*(GPP) of degree n , denoted by (Π, V_o, V_g) :

Given a set of permutations Π , two n -dimensional vectors V_o , and V_g , efficiently determine if $V_g \in G(V_o)$ and if so find a permutation h in the group $G = \langle \Pi \rangle$ such that $h(V_o) = V_g$, and h has a short expansion in terms of Π .

We will take "efficient" and "short" to mean that the solution time and length do not grow too fast with the dimension n . Unless otherwise stated we will implicitly assume that the degree of a GPP is n . We will often

find it convenient to associate a variable (called a *state variable*) with each component of the n -dimensional vectors, called *state vectors*. Then the permutations can be said to permute the values of the n state variables, v_1, v_2, \dots, v_n . Thus, for all i , if v'_i denotes the new value of v_i after applying a permutation $g \in \Pi$, then $v'_i = v_j$ such that $g(j) = i$.

We will be most interested in using the information from Π and V_o to derive information that allows us to quickly solve any GPP of the form (Π, V_o, Y) . This is because by the transitivity of actions this would allow us to quickly realize a solution of any GPP problem of the form (Π, X, Y) (where $X, Y \in G(V_o)$) which is exactly what is needed to plan the actions of an automata. To avoid introducing additional terminology we will use the term GPP to refer to both a triple (Π, V_o, V_g) and a pair (Π, V_o) . The latter style of problem will indicate that we are trying to solve (Π, V_o, Y) for arbitrary Y . We will use this notion of a GPP whenever the goal vector is superfluous. Anything shown for a (Π, V_o) GPP will be true for (Π, V_o, V_g) problems.

4.1 HARDNESS OF GPP

It is natural to wonder how “hard” GPP is. Briefly, we can show (sketches of the proofs are in the appendix):
(a) Determining if $V_g \in G(V_o)$ is no harder than the Setwise Stabilizer Problem¹ (but we have not shown it to be necessarily as hard as this problem).²
(b) Finding $h \in G$ such that $h(V_o) = V_g$ is polynomial time equivalent (with respect to randomized Turing reductions) to the Setwise Stabilizer Problem.

And it is known that determining the length of the shortest expansion of h in terms of Π is P-space complete [FSS⁺89]. It must be emphasized that these are worst case results and may not correspond to the typically observed behavior. Furthermore, these complexity results are all relative to the problem size, n , but it may be to measure complexity in terms of the length of the shortest expansion of h in terms of Π .

4.2 GPP AND PERMUTATION GROUP ALGORITHMS

If the values of the state variables in V_o are all distinct, then all vectors in $G(V_o)$ will have distinct state variables. Thus given a pair of vectors (V_o, V_g) , where V_o contains distinct state variables, the permutation (if any) π_g that maps V_o to V_g is uniquely defined and can be found simply by examining the two vectors. To

¹The Setwise Stabilizer Problem is known to be at least as hard as Graph Isomorphism but is not known to be NP-Complete[Hof82]

²Determining if a partially specified labeling \tilde{V}_g can be extended into a labeling V_g such that $V_g \in G(V_o)$ is easily seen to be NP-complete (reduction from the Clique Problem[GJ79]).

solve the problem, it then only remains to determine a short expansion (if any) of π_g in terms of permutations from Π . We refer to this special case of GPP, where V_o has distinct components, as a *Standard Permutation Problem*, SPP.

This is a classic problem in computational group theory. In [FHL80], Furst, Hopcroft and Luks describe a polynomial-time (in n) algorithm (the FHL-algorithm) to construct a table for a permutation group $G \subseteq S_n$ presented in terms of its generators Π . This table can then be used to find an expansion in terms of Π for any permutation $g \in G$. The *length* of this expansion may be exponentially long in n , but no results exist that guarantee that the expansion length is polynomially related to the optimal expansion. It must be noted that certain generator sets Π exist that generate permutations g whose optimal expansions are exponentially long in n . Clearly, more work needs to be done in this area.

In general, in a GPP, the values of the components of V_o need not be distinct, and instead of a unique permutation π_g that maps V_o to V_g there may be a set $H \subseteq S_n$ of permutations that do the mapping. However, only the permutations in $G \cap H$ are *feasible*, i.e., expandable in terms of Π . The set H may be exponentially large (in n), so a naive approach that checks each $h \in H$ for feasibility is ruled out. Thus one must first find feasible permutations $G \cap H$, before finding a feasible permutation with a short expansion. In section 6 we will show polynomial time algorithms that attempt to transform the original GPP to a smaller GPP which may turn out to be an SPP, so that the existing algorithms for SPP may be used.

4.3 GPP AND SERIAL/TOTAL DECOMPOSABILITY

Consider a variant of the GPP (Π, V_o) where the actions need not be permutations, so that the set of bijections Π is replaced by a finite set \mathcal{F} of functions that map n -dimensional vectors to n -dimensional vectors. Suppose that the functions in \mathcal{F} are, as above, serially decomposable. Such problems (\mathcal{F}, V_o) are called *serially decomposable problems* (SDP). By definition, not every SDP is a GPP. We have shown, in a GPP, the permutations need not be serially decomposable, so not every GPP is an SDP.

In [Kor85], Korf shows an algorithm that constructs a *macro table* (similar to the FHL table) that can be used to solve SDPs of the form (\mathcal{F}, V_o) . The table-construction roughly takes the same amount of time as solving one (\mathcal{F}, V_o, V_g) problem without the macro table (using conventional search techniques). This could take time exponential in n . However, once constructed, this table can be used to quickly generate short solutions for arbitrary V_o . In fact, the solutions are guaranteed to be no longer than n times the opti-

mal solution length. And such solutions can be found in time n times the length of the optimal solution. Korf's Macro Table (KMT) algorithm assumes that the actions are serially decomposable w.r.t. the state variables, and does not attempt to detect whether this property holds, nor does it try to change the representation so that it does. Both these questions must be addressed when trying to apply the KMT algorithm to a GPP. We consider these questions in sections 5 and 7.

The Korf Macro Table has m rows and n columns, where each column corresponds to a state variable, and the rows correspond to the possible values of the variables. Thus the size of the macro table is proportional to the product of the number of variables, and the total number of possible values of the variables. This fact will be relevant in a future section.

In a GPP, the effect of any action is simply a permutation of the state vector. As we show in section 5, we can take advantage of this fact, to devise a probabilistic polynomial-time algorithm to determine if the permutations Π are totally decomposable. If they are, since total decomposability implies serial decomposability, the KMT algorithm can be applied to the GPP. In section 6 it is demonstrated how some GPP problems can be reduced in size. In section 7 we will show that every SPP can be easily transformed to a SDP.

5 DETECTING SERIAL/TOTAL DECOMPOSABILITY

Given a GPP (Π, V_o) , if the permutations Π are serially decomposable w.r.t some ordering on the state variables (i.e., vector-components), the KMT algorithm can be applied to solve problems. However, how can the agent find whether such an ordering exists, and find one if it does?

We show that this problem is reducible to determining whether an arbitrary subset of the state variables is *self-dependent*, in the sense defined below. For any permutation $g \in \Pi$, and any subset T of the state variables, we use the notation $g(T)$ to denote the (list of) values of the variables T after applying permutation g to the initial state V_o . Also gh denotes the product of two permutations (i.e. the result of applying g , then h).

Definition 1 (Self-Dependency) A set of state variables T is self-dependent with respect to permutation $h \in \Pi$ (and labeling V_o) iff the values of the variables T after applying h are a function only of the values of the variables T before applying the action h . That is, there is a function f such that

$$\forall g \in \Pi : g(h(T)) = f(g(T))$$

If a set of variables T is *not* self-dependent, then what

evidence of this exists in the behavior of the environment? This is answered by the following lemma.

Lemma 1 (Witnesses) A set of state variables T is *not* self-dependent w.r.t. permutation $h \in \Pi$ iff there are two permutations g and g' in Π such that

$$g(T) = g'(T) \text{ and } gh(T) \neq g'h(T)$$

We call the two permutations g and g' *witnesses* to the non-self dependency of T w.r.t. permutation h .

We can show the following connection between serial decomposability and self-dependency:

Theorem 1 An permutation $h \in \Pi$ is serially decomposable w.r.t. a set of variables T , iff T can be partitioned into disjoint subsets $\sigma_1, \sigma_2, \dots, \sigma_k$ such that each variable in σ_1 is self-dependent w.r.t. h , and for $i = 2, 3, \dots, k$, each variable x in σ_i has the property that the set of variables

$$\sigma_1 \cup \sigma_2 \dots \cup \sigma_{i-1} \cup \{x\}$$

is self-dependent w.r.t. h .

Sketch of Proof:

If h is serially decomposable w.r.t. an ordering $\langle v_1, v_2, \dots, v_m \rangle$, then the above conditions are satisfied by the partition $\sigma_i = \{v_i\}$. This is easy to show from the definition of serial decomposability. Conversely, if such a partition exists, then it is easy to show that h is serially decomposable when the variables of T are arranged in increasing order of the index i of the partition σ_i to which they belong. \square

This theorem suggests the following algorithm for determining serial decomposability of an action $h \in \Pi$ w.r.t. the n state variables. $\text{self}(\Pi, V_o, T, h)$ is a predicate that tests self-dependency of a set of variables T w.r.t. action h . Let X be the set of the n state variables.

```

procedure Serial ( $\Pi, V_o, h$ )
i  $\leftarrow 0$ ;
 $\sigma_0 \leftarrow \phi$ ;
 $\text{left} \leftarrow \phi$ ;
repeat
     $i \leftarrow i + 1$ ;
     $\sigma_i \leftarrow \{x \in (X \setminus \text{left}) \mid \text{self}(\Pi, V_o, \text{left} \cup \{x\}, h)\}$ ;
     $\text{left} \leftarrow \text{left} \cup \sigma_i$ ;
until  $\text{left} = X$ ;
endproc

```

Central to the above algorithm is the procedure $\text{self}()$, and we show below that this could take time exponential in $|T|$. It must be noted that constructing a KMT by existing weak methods can take time exponential in the diameter of the problem space³ so unless we

³the minimum number of operators needed to transform any state vector in the problem space to any other

dealing with a family of problem spaces where the diameter does not increase as fast as $|T|$ we see that the running time **Serial()** and **self()** is dominated by the time to actually construct a KMT once we have found a set of serially decomposable variables.

By the witness lemma above, to check if a set of variables T is *not* self-dependent, it is sufficient to generate the two witness permutations g and g' in $G = \langle \Pi \rangle$, from the generators Π . Suppose that the agent generates permutations of G by applying a random sequence of actions from Π . Then how likely is it that two such witness permutations will be generated, when T is *not* self-dependent?

To answer this, we use arguments similar to those given in [RS87]. Note that when action (permutation) b is applied to a state vector, the new value of a variable v_i is the current value of the variable v_j where $j = b^{-1}(i)$. As a mnemonic, we use $b^{-1}(v_i)$ to denote the variable v_j . Similarly, $b^{-1}(T)$ denotes the set of variables whose current values become the future values of the variables T after action b . Then the witness property above can be re-stated as: If T is not self dependent w.r.t. action b , then there are two permutations $g, g' \in G$ such that $g(T) = g'(T)$, and $g(b^{-1}(T)) \neq g'(b^{-1}(T))$.

Now consider the permutations of $G = \langle \Pi \rangle$ that map the variables of T and $b^{-1}(T)$ to themselves, i.e., the subgroup H_T that stabilizes $T \cup b^{-1}(T)$. Consider the graph \mathcal{H}_T of left-cosets gH_T of H_T in G . Then, if T is not self-dependent, there are two left cosets gH_T and $g'H_T$ such that the above witness property is satisfied for g, g' . Further, applying a random sequence of actions is equivalent to taking a random walk on the coset graph \mathcal{H}_T , starting from H_T . It can be shown that to ensure that two particular nodes are visited with probability at least $1/2$, it is sufficient to take random steps of length $O(me)$ where m is the number of nodes in \mathcal{H}_T , and e is the number of edges of \mathcal{H}_T . So, to detect self dependency of a set of variables T with a probability of error at most ϵ , the agent need only take a random walk on \mathcal{H}_T whose length is polynomial in m, e , and $1/\epsilon$. After each action, it must check if the witness property holds, and this can done in time polynomial in n , the size of the state vectors.

How large can the coset graph \mathcal{H}_T be? The number of nodes $m \leq \binom{n}{|T|}$, since that is the number of different ways of mapping a given set of $|T|$ nodes to $|T|$ other nodes. Also, $e \leq m|\Pi|$. Thus the number of nodes of the coset graph is $O(n^{|T|})$, exponential in $|T|$. Since T could be $O(n)$ in general, this means the serial decomposability detection algorithm above can take time exponential in n .

However, if the algorithm always checks for the self-dependency of a *constant-sized* set T , then the algorithm runs in time polynomial in n . We can show

easily that a set of variables T is totally decomposable iff every singleton subset of T is self-dependent. Thus a procedure **Total()** can be constructed which simply tests for the self-dependency of each state variable. If an environment passes this test, the agent can apply the KMT algorithm to solve problems.

We have thus shown:

Theorem 2 Given a GPP, (Π, V_o) , of degree n , and a reliability ϵ , the procedure **Total()** runs in time polynomial in $(|\Pi|, n, 1/\epsilon)$ and detects, with error probability $< \epsilon$, whether the permutations Π are totally decomposable.

It is not hard to see that simple adaptations of **Serial** and **Total** can detect decomposability (serial/total) w.r.t small clusters of variables rather than individual variables.

6 REDUCING TO A SMALLER PROBLEM

Because **self()** can run in time exponential in $|T|$ it is important to reduce the size of $|T|$ whenever possible. We now develop a polynomial-time algorithm that attempts to transform a GPP $P = (\Pi, V_o)$ to an equivalent GPP $P' = (\Pi', V'_o)$ of smaller degree. The basic idea is to detect whether the permutations Π permute *blocks* of variables in a certain structured way: if so, then the GPP can be re-cast in terms of block-level permutations, which necessarily have a degree smaller than n .

Two GPPs P, P' are equivalent if:

- (a) there is a one-one correspondence $\phi : \Pi \rightarrow \Pi'$,
- (b) there is a one-one correspondence $\gamma : G(V_o) \rightarrow G'(V'_o)$, where $G = \langle \Pi \rangle, G' = \langle \Pi' \rangle$, and
- (c) For any $V, W \in G(V_o)$ and $\pi \in \Pi$, $\pi(V) = W \Rightarrow \phi(\pi)(\gamma(V)) = \phi(W)$.

Our idea is an extension of the notion of a *closed partition* [Koh78]. Consider a partition τ , and a total order \prec on the index-set $I_n = \{1, 2, \dots, n\}$. We write $\tau(i, j)$ when i, j belong to the same block of τ . Then, w.r.t. a GPP (Π, V_o) , we say that (τ, \prec) is a closed, faithful (CF) partition, if

$$\forall i, j : \forall g \in \Pi : (\tau(i, j) \wedge i \prec j) \Rightarrow \tau(g(i), g(j)) \wedge g(i) \prec g(j)$$

The ordering \prec is called a *faithful ordering*. Note that the blocks of a CF partition must all have the same size. Thus, if the GPP has a CF-partition (τ, \prec) , the effect of any permutation in Π is to map \prec -ordered blocks of τ to \prec -ordered blocks of τ . Further, for any vector in $G(V_o)$ we can then associate a value-pattern with each block, which is obtained by laying out the elements of each block in the order \prec .

Suppose the blocks are numbered $1, 2, \dots, m$, and the value patterns are numbered $1, 2, \dots, k$. Since the par-

tition is faithful, the value patterns in any vector in $G(V_0)$ will be the same, except that their positions will have changed. So for each $g \in \Pi$, we can define a new block-level permutation g' . And for any vector $V \in G(V_0)$, we can define a corresponding m -dimensional vector V' whose i th component gives the value-pattern number of the i th block of τ . Thus we can reduce the GPP (Π, V_0) of degree n to an equivalent, smaller GPP (Π', V'_0) of degree m . We refer to this reduction as a *CF-reduction*.

How do we find a CF-partition? We can do this with a slightly modified version of Kohavi's [Koh78] algorithm, which finds *all* closed partitions of a state graph of an automaton in time polynomial in the size of the state graph. By adding a check to determine faithfulness, this algorithm can be used to find all CF partitions in time polynomial in n , the degree of the GPP. We can choose which CF-partition to use, so that the resulting new GPP has useful properties. For instance, if there is a CF-partition with distinct value-patterns in each block, then we should prefer to use that partition over one which does not have this property. Then the new GPP will be an SPP/SDP, so that both the Korf Macro Table algorithm and the FHL algorithm can be applied.

Example:

These ideas can be illustrated by the Top-Spin puzzle environment, \mathcal{E}_{top} , with the "rotate" (r) and "swap" (s) actions, and binary predicates $p_{ij} = 1$ if position i has disc j , and $p_{ij} = 0$ otherwise. The correctly inferred update graph for this environment should consist of 20 disjoint, isomorphic subgraphs $U_j, j = 1, 2, \dots, 20$, where subgraph U_j contains 20 nodes for the 20 predicates p_{ij} associated with disc j . If we number the nodes in U_j as 1, 2, ..., 20, then the r action defines a cyclic permutation (1 2 ... 20), and the s action defines two cyclic permutations (1 4) and (2 3). In any state, exactly one node p_{ij} in each U_j will have value 1, and all others have value 0. Since each disc is in a different position, the i for which p_{ij} is 1 will be different in each subgraph U_j .

A closed, faithful partition of this update graph is one in which each block i contains the nodes $p_{ij}, j = 1, 2, \dots, 20$, that is *each block corresponds to a position*. The faithful ordering on each block i simply orders the p_{ij} in ascending order of j . Then the new (meta) update graph U' will have 20 (meta) nodes, and each node will have a *distinct* value, since the blocks have distinct value patterns. The s and r actions define cycles which are similar to the original cycles. Thus we can reduce an update graph having 400 nodes with indistinct values to an equivalent update graph with 20 distinct-valued nodes. Thus in addition to reducing the size of the update graph, we have managed to obtain a better-behaved update graph: Any problem on the new update graph is now an SPP, and, as seen before, both the FHL and KMT algorithms apply. Also,

the new update graph in a sense describes the environment more "naturally": The value of the (meta-node) i represents the number of the disc in position i , and the r and s arrows represent how the discs move among the positions when the corresponding actions are performed. ■

The main result of this section can be summarized as:

Theorem 3 *If the index-set $\{1, 2, \dots, n\}$ has a closed, faithful (CF) partition τ relative to the GPP (Π, V_0) of degree n , then this GPP is equivalent to a smaller GPP of degree equal to the number of blocks of τ . Furthermore, all CF-partitions can be found in polynomial time in n .*

7 TRANSFORMING TO A TOTALLY DECOMPOSABLE PROBLEM

We show in this section that it is always possible, at the expense of problem size, to change the representation of the state vectors in a GPP so that the actions (although no longer permutations of the state vectors) are totally decomposable.

For the GPP of degree n , (Π, V_0) , Consider the following *inverted* representation X_V of an n -dimensional vector V : $X_V = \langle x_1 x_2 \dots x_k \rangle$ where x_i is the set of positions (node indices) that have value t_i . The number of occurrences of the k different values must remain the same after any action; the values simply get rearranged in the vector. We can show easily that

Lemma 2 *In the inverted representation X_V , all actions are totally decomposable.*

Proof:

Whenever an action b is applied to the update graph, the values t_i move according to the permutation π_b . Thus the indices which acquire value t_i after this action depend only on which indices currently have the value t_i . Thus the new value of x_i depends only on its current value.

□

Since total decomposability implies serial decomposability, the KMT approach could conceivably be used with this representation, although the macro table can in general be very large: the number of rows in the table is equal to the size of the set of possible values of the variables, and this set could be large. For example, the number of possible values of x_i is $\binom{n}{n_i}$, where $n = |C|$, the number of nodes in the update

graph. However, if the n_i are all bounded by a constant c , this number would be polynomial in n , which means the macro table will be of size polynomial in n , and so problems can still be solved in time polynomial in the length of the optimal solution.

Clearly, when all the $n_i = 1$, we have a standard permutation problem, SPP, so we have:

Lemma 3 *Every SPP can be transformed to a SDP by inverting the state vector representation.*

Thus, for an SPP, both the FHL algorithm and the KMT algorithm are applicable.

Example:

The ideas of this section can be illustrated by the example of the Hungarian Rings puzzle: This is a pair of interlocking rings of beads. The two rings have two beads in common where they intersect, and the beads in each ring can be rotated on that ring. The beads are colored with k different colors, and there are m of each color. The goal is to achieve a given color-configuration. Consider the environment \mathcal{E}_{hun} corresponding to this puzzle, with predicates p_i giving the color of the bead in the position numbered i . This environment is a permutation environment, and its update graph will have km nodes, and the values of the nodes will reflect the colors of the beads in the puzzle. There are thus m nodes of each color. Now if we take the inverse representation, there are k variables $x_1, x_2 \dots x_k$, corresponding to the k colors. The value of each variable x_i is the set of positions containing beads of color i . Under this representation, all the actions are totally decomposable, so the KMT algorithm can be applied.

The number of possible values of the variables is $\binom{mk}{m}$. So if we consider the class of Hungarian Ring puzzles with constant m and variable k , then the size of the Korf macro table is polynomial in k . ■

8 A RECIPE FOR SOLVING PROBLEMS WITH AN UPDATE GRAPH

There are many ways in which the different ideas presented in this paper can be combined into a strategy for attacking GPPs. We present here one possible recipe for solving a GPP (Π, V_o) .

Procedure GPP-Solve(Π, V_o)

1. $P \leftarrow (\Pi, V_o)$;
2. If V_o has distinct values then
 - P is an SPP, so
 - apply FHL to P , or
 - apply KMT to the inverted representation of P .
 - return "success"
- endif

3. Find all CF-reductions P' of P
 - If there is a CF-reduction $P' = (\Pi', V'_o)$ then
 - such that V'_o has distinct values,
 - $P \leftarrow P'$
 - goto 2.
 - endif
 4. Find the smallest CF-reduction $P' = (\Pi', V'_o)$ of P .
 - If $\text{degree}(P') < \text{degree}(P)$ then
 - $P \leftarrow P'$
 - goto 2.
 - endif
 5. If P is totally decomposable then
 - apply KMT to P .
 - return "success"
 6. (All simplifications and reductions fail)
 - will have to use search methods to solve instances of (Π, V_o, V_g) individually, couldn't solve (Π, V_o)
 - return "failure"
- endproc

9 CONCLUSION

The fundamental problem in integrating a model-learning algorithm with a problem-solving algorithm is that the problem-solving algorithm may make assumptions about the state representation that are not always satisfied by the learned internal model. In this paper, we considered the problem of coupling Rivest and Schapire's model-learning algorithm with two problem-solving algorithms: Korf's Macro Table (KMT) algorithm and the permutation group algorithm of Furst, Hopcroft, and Luks (FHL). If the inferred representation (an update graph) is totally decomposable, the KMT algorithm can be applied. We presented a polynomial-time algorithm that detects whether the learned update graph satisfies this property. We also presented a representation-change that transforms an update graph problem into a totally decomposable problem. This same transformation can sometimes change the problem into a permutation group problem that can be handled by the FHL algorithm. Finally, we presented a polynomial-time algorithm that finds possible reductions of the update graph to a smaller, equivalent update graph. In addition to being smaller (and therefore more tractable) the reduced update graph may satisfy the requirements of the above two algorithms.

Though we have shown how to recognize when a set of variables in an update graph is serially decomposable, detecting serial decomposability in time polynomial in the size of the update graph remains an open problem. Since there exist problems whose optimal solutions are exponentially long in the size of the update graph, it is more practical, from an AI perspective, to measure efficiency relative to the length of the optim-

mal solution. Developing practical algorithms which are provably efficient in this sense is an open research area. Also, methods need to be developed for analyzing the computational complexity of problem-solving algorithms relative to this measure.

10 ACKNOWLEDGEMENTS

We would like to thank Merrick Furst, Ravi Kannan, Craig Knoblock, Gary Miller, Tom Mitchell and Ron Rivest for helpful discussions.

A HARDNESS PROOFS

In this section we consider the GPP (Π, V_o, V_g) and $G = \langle \Pi \rangle$. From [Hof82] we know that the Setwise Stabilizer Problem and the Coset Intersection Emptiness Problem problems for groups are polynomial time equivalent. So we will use them interchangeably. Some of these theorems should be known, and we are presenting them here only for completeness.

Theorem 4 *Determining if $V_g \in G(V_o)$ can be solved by solving a Coset Intersection Emptiness Problem.*

Proof:

Let π be any permutation in S_n such that $\pi(V_o) = V_g$ (it is trivial to determine if such a permutation exists and if so to find it, if no such permutation exists then $V_g \notin G(V_o)$). Let H be the group of all permutations in S_n that fix V_o (it is easy to find generators for H). Then $V_g \in G(V_o)$ iff $H\pi \cap G \neq \emptyset$.

□

Lemma 4 *Given $\pi \in S_n$ and $G, H \subset S_n$ such that $H\pi \cap G \neq \emptyset$ finding $h \in H\pi \cap G$ is poly time equivalent to the Coset Intersection Emptiness Problem.*

Proof:

This can be done by noticing that the FHL table for G (which can be constructed from generators in polynomial time) yields (by deleting initial rows and columns) a subgroup tower $I = G^{(n)} \subset G^{(n-1)} \subset \dots \subset G^{(0)} = G$ such that the index of $G^{(i+1)}$ in $G^{(i)}$ is no greater than n . Let k be the largest index such that $H\pi \cap G^{(k)} \neq \emptyset$. WLOG we can assume $k = n$ (else we can replace G by $G^{(k)}$). Let $G' = G^{(n-1)}$ and $\tau_1, \tau_2, \dots, \tau_k$ ($k \leq n$) be a complete right traversal of G by G' . Then since $G = \cup_{i=1}^k \tau_i G'$ we must have $H\pi \cap G' \tau_j \neq \emptyset$ for some j such that

$1 \leq j \leq k$. We can check these k possibilities by checking Coset Intersection Emptiness on $H(\pi\tau_i^{-1}) \cap G'$ for $i = 1 \dots k$. So we (recursively) solve the smaller problem by finding $h' \in H(\pi\tau_j^{-1}) \cap G'$ and $h = h'\tau_j$ is the answer.

□

Theorem 5 *Finding $h \in G$ such that $h(V_o) = V_g$ is polynomial time equivalent (with respect to randomized Turing reductions) to the Setwise Stabilizer Problem.*

Proof:

From the previous proof it is easy to see that to find $h \in G$ such that $h(V_o) = V_g$ all we have to do is find $h \in H\pi \cap G$ (when $H\pi \cap G \neq \emptyset$) which, by the previous lemma, is poly time equivalent to Coset Intersection Emptiness (and hence Setwise Stabilizer).

For the other direction let G be an arbitrary subgroup of S_n , Π a set of generators for G and $P \subset \{1, 2, \dots, n\}$ arbitrary. We will use the GPP problem to calculate the Setwise Stabilizer of P in G . Let V_o be the length n vector $(V_o)_i = 1$ if $i \in P$ else $(V_o)_i = 0$. Assume we can solve GPP instances (Π, V_o, V_g) . We can calculate the setwise stabilizer S of P in polynomial time with exponentially small chance of error by running the following procedure:

1. $S = \{\text{identity}\}$
2. Repeat n^2 times:

Let π be a random permutation from G .
Let h be a solution of $(\Pi, V_o, \pi(V_o))$.
Let $S = \langle S, \pi h^{-1} \rangle$.
endrep

The procedure works because during each iteration S is a subgroup of $\text{stab}(P, G)$ therefore if $S \neq \text{stab}(P, G)$ we have $|S| \leq |\text{stab}(P, G)|/2$. The GPP solution procedure has no way of knowing π so since π is randomly selected from G we see that πh^{-1} will be randomly selected from $\text{stab}(P, G)$, so if $S \neq \text{stab}(P, G)$ we know $\pi h^{-1} \notin S$ with probability $\geq 1/2$. and in addition we will have $|\langle S, \pi h^{-1} \rangle| \geq 2|S|$. Therefore we need only about $\log_2 n!$ successes to compute $\text{stab}(P, G)$. $\log n!$ is asymptotically equal to $n \log n$ and is therefore dominated by n^2 , so it is a simple matter of statistics (Binomial trials) to show our chance of failure is exponentially small.

□

Theorem 6 Deciding if there is an extension of a partial labeling \tilde{V}_g into a labeling V_g such that $V_g \in G(V_o)$ is NP-complete.

Proof:

Since a labeling V_g and a permutation h can be presented succinctly and we can check in poly time if $h \in G$ and $h(V_o) = h(V_g)$ it is clear that this problem is in NP.

To prove it is NP-complete we demonstrate that this problem can encode the Clique problem (determining if a graph has a clique of size k , which is known to be NP-complete). Let K be an arbitrary graph of n nodes encoded as a length n^2 vector V_o such that $(V_o)_{n*(i-1)+j} = 1$ if there is an edge from node i to node j and $(V_o)_{n*(i-1)+j} = 0$ otherwise. Let G be S_n acting on V_o such that for $\pi \in S_n$ and a length n^2 vector v we have: $(\pi(v))_{n*(i-1)+j} = v_{n*(\pi(i)-1)+\pi(j)}$. Let \tilde{V}_g be the length n^2 vector with 1's in the first k^2 entries and the rest undetermined. Then there exists V_g such that V_g agrees with \tilde{V}_g in all the specified positions and $V_g \in G(V_o)$ if and only if K has a k -clique.

□

References

- [FHL80] M.L. Furst, J.E. Hopcroft, and E. Luks. “Polynomial-time algorithms for permutation groups” In *Proc. 21st IEEE Foundations of Computer Science*, pages 36–41, 1980.
- [FSS⁺89] A. Fiat, M. Shahar, A. Shamir, I. Shimshoni, and G. Tardos. “Planning and learning in permutation groups” In *Proc. 30th IEEE Foundations of Computer Science*, pages 274–279, 1989.
- [GJ79] M. Garey, and D. Johnson. *Computers and Intractability A Guide to the Theory of NP-Completeness* W.H. Freeman and Company, 1979.
- [Hof82] Christoph M. Hoffmann. *Group-Theoretic Algorithms and Graph Isomorphism* Springer-Verlag, 1982.
- [Koh78] Zvi Kohavi. *Switching and Finite Automata Theory*. Mc Graw-Hill, 1978.
- [Kor85] R.E. Korf. “Macro-operators: A weak method for learning” *Artificial Intelligence*, 26:35–77, 1985.
- [Mac68] I.D. Macdonald. *The Theory of Groups*. Oxford University Press, 1968.
- [RS87] R. L. Rivest and R. E. Schapire. “Diversity-based inference of finite automata” In *Proceedings of the Twenty-Eighth Annual Symposium on Foundations of Computer Science*, pages 78–87, Oct 1987.
- [Sch88] R.E. Schapire. *Diversity-based inference of finite automata* Master’s thesis, Massachusetts Institute of Technology, May 1988.

Immortal: a Model-based Belief Revision System

Timothy S-C Chou and Marianne Winslett

Department of Computer Science

University of Illinois

Urbana, IL 61801

chou@cs.uiuc.edu

Abstract

Recently the AI community has seen a flurry of proposals for model-based semantics for updating knowledge bases expressed as logical theories. It is not at all clear, however, how to translate a proposed model-based semantics into a procedure suitable for implementation. As a result, it becomes hard for one to analyze and identify the practical limits of belief revision semantics for solving large interesting problems. In this paper, we propose an architecture for implementing model-based theory revision semantics, discuss the major algorithms implemented in our prototype system, analyze the complexity of the algorithms, and report some results from the implementation of the algorithms. The key feature of our algorithms, as shown in the complexity analysis, is that while belief revision is intractable in general, the expected running time of our algorithms will be polynomial for many interesting cases.

1 Introduction

Belief revision deals with the issue of updating one's state of belief with newly received information, in particular when the new information conflicts with the current state of belief [Abiteboul 85, Alchourrón 85, Dalal 88, Forbus 89, Gärdenfors 88ab, Hegner 87, Katsumoto 89, Rao 89, Reiter 87, Sandewall 89, Satoh 88, Weber 86, Winslett 88]. The belief revision semantics that appear in the literature seem promising for defining the meaning of updates to knowledge bases expressed as logical theories. Unfortunately, these semantics are mainly theoretical tools and have not yet seen practical use, with the exception of [Forbus 89], who used pre-analysis to predetermine the effects of all revision requests that could be given to his system. It is not at all clear whether it is possible to translate these semantics into practical implementa-

tions that do not rely entirely on pre-analysis. The lack of implementation efforts prevents one from

- knowing how model-based belief revision can be accomplished in practice;
- working with various semantics and justifying their practical use; and
- theoretically analyzing and identifying the practical limits of revision semantics for solving large interesting problems.

The last point mentioned above is important in that while belief revision is intractable in theory, we expect certain problems can be solved within some reasonable bound, as is true in many technical areas such as linear programming.

In this paper, we present a general architecture of a model-based belief revision system. We then discuss the major algorithms implemented in Immortal (an implementation of model-based belief revision system), which is a prototype system for the study of various belief revision semantics, in particular, those similar to the Possible Models Approach (PMA) [Winslett 88]. The complexity of the algorithms will be analyzed, and some preliminary results will be reported.

2 The Targeted Types of Knowledge Bases and Belief Revision Semantics

Immortal is built for use with first-order knowledge bases that employ a Herbrand universe assumption and have a finite, fixed set of relevant terms¹ (essentially propositional knowledge bases). Updates are insertions of finite sets of ground (i.e., variable-free) formulas. The knowledge base to be updated (hereafter called the theory) is assumed to be composed of a finite

¹Our next direction of extension for Immortal is to remove the Herbrand universe assumption and handle updates involving equality and infinite sets of relevant terms, in the case where the knowledge base contains only universal formulas.

set P of non-updatable formulas, which represent fixed knowledge about the problem domain; and a finite set of updatable formulas. The belief revision semantics to be employed by Immortal has the following characteristics, when formula ω is inserted into theory T :

1. Consider a particular model M of T . The result of updating M is the subset of $\text{Models}(\{\omega\} \cup P)$, the set of all models of ω and P , such that the members of the subset are closest to M under some measure of distance.
2. The set of models produced by updating T is the union of the sets of models produced by updating the models of T .

Such semantics include those of [Hegner 87, Reiter 87, Winslett 88]. Immortal can be tailored to a particular choice of semantics from this class by changing one particular routine, as described below. Immortal is also well suited (i.e., requires only a one-line change in the algorithms) for two other closely related families of semantics: those adopting the heuristic that when ω is consistent with T , the result of the update is $\text{Models}(T \cup \omega)$ (e.g., [Alchourrón 85, Borgida 85, Gärdenfors 88a]); and those where the result models of the update are the members of $\text{Models}(\{\omega\} \cup P)$ that are closest to some member of $\text{Models}(T)$ (e.g., [Dalal 88, Satoh 88]).

The measure of distance mentioned in item 1 is used to prune out unlikely result models. For AI applications it is often helpful to assign priority levels to the predicates of the language, so that higher priority predicates are given more weight when measuring the distance between models; the architecture of Immortal includes direct support for priorities on predicates when the set of *minimal inconsistencies* is *complete* (defined below) at each priority level.

The semantics used in experimental runs of Immortal and in our complexity analysis is that of the prioritized Possible Models Approach (PPMA) [Winslett 88]: Let a model of T be a set of literals, such that if the literals in the set are assigned the truth value *true*, and those not in the set are assigned the truth value *false*, all the formulas of T are satisfied under that truth assignment. Let the difference between a pair of models M and N , $\text{Diff}(M, N)$, be the set of atoms that occur in only one of the two models. A model N is closer to M than N' if there is some priority level i such that

1. $\text{literals}(i) \cap \text{Diff}(M, N)$ is a proper subset of $\text{literals}(i) \cap \text{Diff}(M, N')$, where $\text{literals}(i)$ is the set of all literals over predicates at priority level i and their negations; and
2. $(\text{literals}(j) \cap \text{Diff}(M, N)) = (\text{literals}(j) \cap \text{Diff}(M, N'))$, for all priority levels $j < i$.

In an application, we would expect the chosen domain-independent update semantics to be augmented with

additional domain-dependent measures of distance, a la [Forbus 89], to give additional pruning of unlikely result models.

3 Definitions and Terminology

An *inconsistency* is a set of literals whose conjunction is inconsistent with P . A set I of *minimal inconsistencies* is one in which if j is a member of I , then no proper subset of j is also a member of I . An inconsistency j follows from a set I of inconsistencies if $\neg I \models \neg j$, where $\neg I$ is the set of sentences containing, for each member m of I , the negation of the conjunction of the members of m , i.e., $\neg m$. A set of minimal inconsistencies I is *complete* if for every inconsistency j , j follows from I . A set of minimal inconsistencies I is *complete at priority level i* if for every inconsistency j whose predicate symbols are all at priority level i or higher, j follows from I' , where I' is the restriction of I to inconsistencies whose members all have priority i or higher.

The set L of *literals of interest* from the language of the application is the set of all legal literals which are formed from the predicate symbols and ground terms of the language and obey the type axioms of the language.

A *model* (of a finite set T of formulas) is a maximum subset of L which satisfies T and contains no complementary literals. A *partial model* is a subset of a model.

Given a set P of non-updatable formulas of a theory T , a model M of T , and an update ω of M , a *result model* is a model satisfying both P and ω . An *intermediate result model* is a result model which is not minimally changed from M under the measure of distance used in the update semantics, whereas a *final result model* is a result model which is minimally changed from M . A *candidate result model* is a model satisfying ω but not necessarily P .

Given a set S of sets of literals, a *minimal selection* s is a set of literals containing one literal from each element of S , such that no proper subset of s has this property. A *flipping* of S is a set containing the negation of every literal in some minimal selection of S .

A *prioritization scheme* is a function which returns the priority level, an integer, of a given predicate symbol.

4 Model Generation

In this section we quickly sketch Immortal's approach to generating models. Given a set S of literals, Immortal is often asked to change S as little as possible (under the measure of distance being employed) to get a model of $P \cup \{\omega\}$. Usually some of the literals in S are non-updatable, meaning that they must be true

in the model(s) Immortal returns; others are updatable, meaning that Immortal can negate their truth assignments (flip the literals) if necessary in order to find a model of $P \cup \{\omega\}$. Immortal tries to convert S to a model of $P \cup \{\omega\}$ by first looking to see how S is inconsistent with $P \cup \{\omega\}$, and then removing those inconsistencies if possible. The set of minimal inconsistencies I between P and L , which is pre-computed, is referenced in this process, as discussed below.

We chose our approach to model generation for the sake of efficiency, based on the assumption that a model will not change drastically when it is updated, in the sense that the changes implied by ω will not have a long chain of ripple effects through M . We highlight how result models are computed; models of the initial theory can be obtained in a similar fashion. To compute the result models from updating, we start by modifying M . More specifically, we convert the insertion formula ω into an initial set of changes to M , incorporate these changes into M , and find the subset of minimal inconsistencies appearing in the resulting model M' due to these changes. A literal from each of the members of the subset is then selected and negated, if possible, to form a *flipping*, which is checked for minimality (i.e., would any proper subset still constitute a flipping?) and then treated as a new set of changes to be applied to M' . This process of finding a subset of inconsistencies, making flippsings, and incorporating changes repeats until consistent result models are found; the process is then repeated for the next truth assignment satisfying ω , and so on until we are finished with M . Shown in Figure 1 is an example where $M = \{A, \neg B, \neg C, D, E\}$ and $\omega = \{\neg A\}$. The set of literals is $\{A, B, C, D, E, \neg A, \neg B, \neg C, \neg D, \neg E\}$. The set I of minimal inconsistencies is $\{\neg A \neg C, AC, \neg B \neg D, BD, \neg A \neg BE, A \neg E, B \neg E, AB\}$ corresponding to the set of non-updatable formulas $\{(A \vee C) \wedge \neg(A \wedge C), (B \vee D) \wedge \neg(B \wedge D), E \leftrightarrow (A \vee B), \neg A \vee \neg B\}$. The final result models are $\{\neg A, B, C, \neg D, E\}$ and $\{\neg A, \neg B, C, D, \neg E\}$.

During the process of model generation, Immortal takes every possible opportunity of noticing that the model being generated in fact will not be minimally changed from M ; when this occurs, Immortal prunes out that search path and backtracks to look for another result model. The choice of appropriate pruning heuristics is crucial in order to avoid combinatorial explosion in the search for result models. The pruning heuristics in Immortal are based on the idea that the measure of distance will involve a test for subset inclusion of changes made from M , as is done in the PPMA. We have found that the use of priorities greatly reduces the computational effort needed to find result models; this is because Immortal generates result models in stages, one stage for each priority level, and the earlier and more frequent feedback on whether a change will be minimal leads to a reduced branching factor in the search tree associated with Immortal's search for

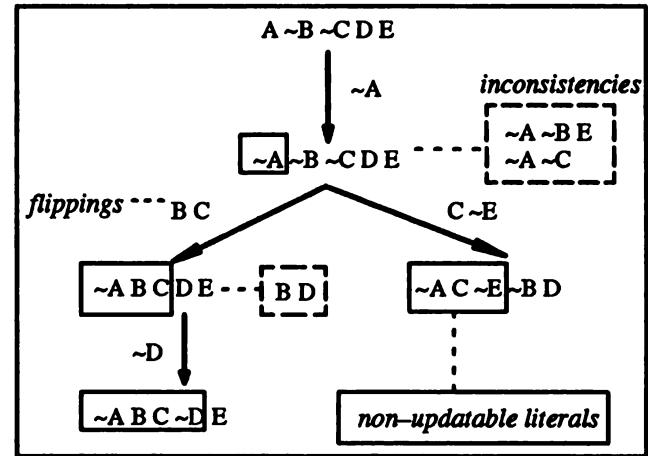


Figure 1: An Example of Model Generation

result models.

5 Architecture

The model-based belief revision system of Immortal is composed of the following components:

1. An inconsistency manager, which provides an efficient interface for storing and accessing the set of minimal inconsistencies;
2. A model generator for computing the initial models of a theory; and
3. A result models finder, which searches for result models under the guidance of some chosen model-based revision semantics and pruning heuristics. Only this component is specific to the particular choice of revision semantics.

We will briefly summarize the operation of the system below to show how these components fit together.

As mentioned before, the set of minimal inconsistencies between the non-updatable formulas of the theory and the set of literals of interest from the language of the application is pre-computed. Immortal does not require a maximal set of minimal inconsistencies; instead, the set of minimal inconsistencies need only be complete, as defined in Section 3. When priorities are used, the set of minimal inconsistencies must be complete at each priority level. These inconsistencies are stored in a database maintained by the inconsistency manager for subsequent reference during update processing.

The minimal inconsistencies can be computed by using a general-purpose theorem prover, any suitable inference engine, or even a special-purpose generation procedure. For our experiments, we used both a theorem

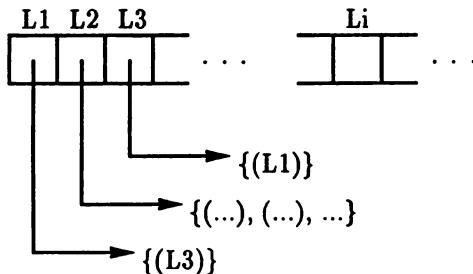
prover and some generation procedures. The theorem prover that we used is a general-purpose first-order resolution-based theorem prover, FRAPPS [Mitchell 89], which was developed at the University of Illinois for use in courses on mechanized inference. FRAPPS is also used during processing of certain updates where *temporary* inconsistencies are needed. We will show in Section 6 how we use FRAPPS to generate these inconsistencies.

The set I of minimal inconsistencies found must be stored by the inconsistency manager in such a way that the following functions can be efficiently supported:

1. For two finite sets of literals L_1 and L_2 , each of which is consistent with P , determine if $\exists i \in I$ such that $i \subseteq (L_1 \cup L_2)$.
2. For two finite sets of literals L_1 and L_2 , each of which is consistent with P , return all inconsistencies in $L_1 \cup L_2$, i.e. $\{i \mid i \in I \wedge i \subseteq (L_1 \cup L_2)\}$.

In an earlier version of the implementation, an ATMS [de Kleer 86] was chosen to manage these inconsistencies. Each literal was represented in the ATMS by an *assumption node* and the inconsistency among a set of literals was an ATMS *nogood* among the nodes corresponding to the literals. We found ATMSs (the ones available to us) were inefficient, however, in that (1) when justifications are used, the propagation and the proliferation of nogoods, which is equivalent to the computation of the closure of the minimal set of inconsistencies, was prohibitively expensive, and (2) when avoiding the use of justifications, it became very time consuming to compute the set of nogoods contained in a given set of literals.

We resort to a database technique to handle the inconsistencies in the current implementation. Specifically, an (inverted) index on I was built. The index, as shown diagrammatically below, is an array of pointers, one for each of the literals of interest, each pointing to a complete subset of inconsistencies where the corresponding literal is common to all elements in this subset. With the index, the subset of I containing any particular literal can be found by a table look-up, and consequently the two needed functions can be efficiently supported.



During update processing, the model generator is used to find each model M of the initial theory, if those

models have not previously been identified. For each model M found, the result models finder is employed to search for the result models obtained by updating M . The inconsistencies maintained by the inconsistency manager are heavily used in both cases.

The search process in the result models finder is guided by the belief revision semantics chosen, e.g., minimal change semantics and prioritization in PPMA. The choice of appropriate pruning heuristics is also crucial in order to avoid combinatorial explosion in the search for result models.

The algorithms for computing models and result models will be detailed in the next section.

6 Algorithms and Procedures

Presented here are the major algorithms and procedures of Immortal including

1. a driver routine for the system, showing how individual procedures are pieced together;
2. an inconsistencies generation procedure (*INCON*);
3. a prioritization routine (*Prioritize*), which is a driver for *RModels*;
4. a result models generation algorithm (*RModels*), which finds result models with the help of *MinimalChange*;
5. *MinimalChange*, which implements the minimal change semantics of PMA; and
6. a model generator (*MODELS*), which is a variant of *RModels*.

We begin by presenting the main system driver.

Let us define two global data structures before presenting the algorithms. The set of pre-computed minimal inconsistencies is stored in I . The query literal q is a dummy literal which is made a member of each of the temporary inconsistencies discussed later. q does not occur in the set of literals of interest.

The Main System Driver

Shown in Figure 2 is the main program which drives the other routines of the system. Updates are taken one at a time. The procedure is self-explanatory.

The Inconsistencies Generation Procedure

Presented in Figure 3 is *INCON*, which finds inconsistencies. *INCON* takes a set L of literals and a set A of formulas, and finds all minimal subsets of L that are inconsistent with A . An inconsistency simply means that the constituent literals cannot all hold in any model composed of literals from L satisfying

Steps:

1. Accept from the user a finite set of literals of interest L , and a prioritization scheme PS covering all the predicate symbols.
2. Let $NewT = True$. /* signal for a new set of updatable formulas */

Loop forever

If $NewT$

- Accept a finite set T of ground updatable formulas.
- If any member of T is not a literal
 - Call $INCON(L, T)$. /* compute temporary inconsistencies */
 - Let $T = \{q\}$.
- Let $M = MODELS(L, L, T)$. /* compute the initial set of models */
- Remove q , if it exists, from m for each $m \in M$, and
 - discard the temporary inconsistencies (those having q as a member), if any, from I .
- Let $NewT = False$.

Accept an update A , a finite set of ground formulas.

Let $ResultModels = \bigcup_{m \in M} Prioritize(m, A)$.

Print $ResultModels$.

Accept *Intention*.

Case *Intention* is

- Done*: Exit.
- NewT*: Let $NewT = True$.
- Otherwise: Let $M = ResultModels$.

Figure 2: The Main System Driver

Input: A finite set of literals L and a finite set of formulas A .

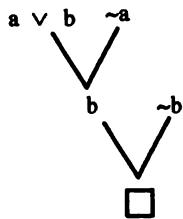
Output: Augment I with the set I_t of minimal inconsistencies between L and A .

Steps:

1. Assert L and A into the theorem prover's database.
 2. /* Find all (resolution) proofs of inconsistencies, i.e. those proofs leading to empty clauses, of minimal length among the clauses in the prover's database. Each of these proofs is represented as a (resolution) proof tree, composed of a collection of leaf nodes and internal nodes, with an empty clause as root. The leaf nodes are either literals in L or formulas in A and the internal nodes are resolvents. The set of inconsistencies can be directly read off from the proofs. */
- Let T be the set of minimal length proofs of inconsistencies among the literals in L and the formulas in A .
- Let $I_t = \{i \mid \exists t \in T \text{ such that } i \text{ is the subset of } L \text{ whose members are labels of leaf nodes of } t\}$.
3. /* Remove non-minimal inconsistencies and then augment I */
- Let $I_t = I_t - \{i \in I_t \mid \exists j \in I_t \wedge i \subset j\}$.
- Let $I = I \cup \{i \cup \{q\}\}$ for each $i \in I_t$, where q is the query literal.

Figure 3: The Inconsistencies Generation Procedure ($INCON$)

A. The set of inconsistencies between L and A can be found using an inference engine, e.g., FRAPPS. Given L and A , the theorem prover can be invoked to find all minimum-length proofs of inconsistencies, i.e., those proofs leading to empty clauses. And the needed inconsistencies can be directly read off from the proofs so obtained. For instance, let $L = \{a, \neg a, b, \neg b\}$ and $A = \{\{a \vee b\}\}$; then the proof shown below (in addition to the proofs of a and $\neg a$, and b and $\neg b$) is obtained, giving the set of inconsistencies $\{\{\neg a, \neg b\}\}$, i.e., one cannot believe both $\neg a$ and $\neg b$ at the same time, given $a \vee b$.



Immortal uses *INCON* to compute *temporary inconsistencies*, which are those discovered when processing a particular set of updatable formulas; temporary inconsistencies are to be stored in I only until the current update has been performed, and are then to be discarded. *INCON* can also be used during the pre-computation of minimal inconsistencies (I) but the process can be very slow if the non-updatable formulas are complicated.

The Prioritization Algorithm

We next present *Prioritize*, shown in Figure 4, a driver routine with built-in prioritization for the generation of result models. Given an initial model M and an update A , *Prioritize* returns a set of models which satisfy A and are closest to M under the semantic measure of distance defined.

As a reflection of prioritization, *Prioritize* generates result models in stages, starting at the highest possible priority level and working down to the lowest. The highest possible priority level is equal to the lowest priority level among those of the predicate symbols appearing in the update A . The algorithm used is the same at each priority level, so we will describe its workings for one particular priority level.

Given a (partial) candidate result model, N , produced from the previous priority level or the given update itself initially, *Prioritize* augments N with the *updatable literals* at the current priority level, giving a new (partial) candidate result model *CandModel*. Updatable literals are those which appear in the initial model but not the update. *CandModel* will then be fed to *RModels*, which searches for ways to make *CandModel* consistent, resulting in a set of (partial) result models deposited in a global variable *Result*. A (partial) result model can be placed in *Result*, only to be pruned

out later in the search when a less-changed (partial) result model is discovered. When done, the (partial) result models in *Result* will become the candidates for revision at the next priority level, if any.

The Result Models Generation Algorithm

RModels is the workhorse of the search for result models. Given a (partial) candidate result model N , which might not be consistent, *RModels* searches for possible ways to make N consistent. The conceptual search tree that *RModels* works on has the following properties. Each internal node in the search tree represents a (partial) candidate result model. The branches from a node represent the set of consistent flippings formed from the subset of minimal inconsistencies appearing in the candidate result model N' represented by this node; these flippings are applied to N' in order to eliminate the inconsistencies found in N' . These branches may lead to different outcomes. Some may lead to (partial) result models, which are stored in *Result*. Others may end up with the situation where no consistent model can be found; these branches are removed. Still others may lead to candidates which are further from M , under the semantic measure of distance, than a previously found (partial) result model and are therefore pruned away. *RModels* traverses the search tree in a depth-first fashion.

The algorithm of *RModels* is given in Figure 5. Step 1 finds the inconsistencies in the given candidate result model N ; these inconsistencies must all have been caused by the *changes* applied to N . Step 2 removes *non-updatable literals*, i.e. those not in the set of updatable literals, from the inconsistencies obtained in step 1 because non-updatable literals cannot be changed (negated). Step 3 computes the set of flippings from the inconsistencies obtained in step 2. For instance, the set of flippings for the inconsistencies $\{\{a, b\}, \{c, d\}\}$ is $\{\{\neg a, \neg c\}, \{\neg a, \neg d\}, \{\neg b, \neg c\}, \{\neg b, \neg d\}\}$, and the set of flippings for $\{\{a, b\}, \{a, c\}, \{b, d\}\}$ is $\{\{\neg a, \neg d\}, \{\neg b, \neg c\}\}$. Each of these flippings will be checked to see if it is consistent with the non-updatable formulas of the theory. Only consistent flippings will be applied to N , and in effect, a new level of branches is added to the search tree. Lastly, step 4 checks to see if N is a final result model with the help of *MinimalChange*.

An example will make this clearer. The search trees of two revision cases are shown below. The PPMA semantics is used in both cases. The set of literals *Changes* is capitalized. Non-updatable literals are placed in boxes. In the first case, we assume there is only one priority level; $L = \{a, \neg a, b, \neg b\}$; the initial model $M = \{\neg a, \neg b\}$; $I = \{a \neg b, b \neg a\}$ corresponding to the set of non-updatable formulas $\{(a \wedge b) \vee (\neg a \wedge \neg b)\}$; and the update A is $\{a \vee b\}$. Since A contains a formula, A will become $\{q\}$ and a temporary inconsistency $\{q, \neg a, \neg b\}$ will be added to I . As shown, the

Input: A model M to be revised and an update A .
Output: The set of models of A that are closest to M under the semantic measure of distance defined.
Steps:

1. Let $StartPriority$, a global variable, be the lowest priority level of those of the predicate symbols in A .
2. If any member of A is not a literal, call $INCON(L, A)$ and let A be $\{q\}$.
3. Let $CandResModels = \{A\}$.
 Let $UpdLits = M - (A \cup \neg A)$, the set of updatable literals in M , where $\neg A$ contains the negation of each of the literals in A .
4. For $p = StartPriority$ down to min(priority levels in PS) do
 - Let $pUpdLits = \{l \mid l \in UpdLits \wedge l's predicate symbol has priority p or higher\}$
 - Let $UpdLits = UpdLits - pUpdLits$
 - Let $Acc = \{\}$
 - For each $N \in CandResModels$ do
 - Let $Result = \{\}$ /* a global variable */
 - Let $Changes = N - M$
 - Let $CandModel = N \cup pUpdLits$
 - Call $RModels(CandModel, pUpdLits, Changes, M, p)$
 - Let $Acc = Acc \cup Result$
 - Let $CandResModels = Acc$
5. Remove the dummy query literal q , if any, from each member of $CandResModels$, discard the temporary inconsistencies (those having q as a member), if any, from I , and return the new value of $CandResModels$.

Figure 4: The Prioritization Algorithm (*Prioritize*)

Input: A partial candidate result model N , a set $UpdLits$ of updatable literals in N , a finite set $Changes$ of literals applied to N , the initial model M , and the current priority level p .

Output: Stores in $Result$ those partial result models of the update that are minimally changed from M at level p according to the semantics defined in $MinimalChange$.

Steps:

1. /* Find the inconsistencies in N . */
 - Let $Incons = \{i \mid i \in I \wedge (i \cap Changes \neq \emptyset) \wedge i \subseteq N\}$
2. /* Remove non-updatable literals, those not in $UpdLits$. */
 - If $Incons \neq \{\}$
 - For all $i \in Incons$
 - Let $i = i \cap UpdLits$
 - If $i = \{\}$
 - return /* inconsistency cannot be resolved */
 3. /* Compute flippings and check for consistency */
 - Let $ChangeSet = \{C \mid C \text{ contains one element from each member of } Incons \text{ and } C \text{ is minimal, i.e., no proper subset of } C \text{ has this property and } \neg C \text{ is consistent, i.e., } \forall c \subseteq \neg C \mid c \in I\}$
 - For each $C \in ChangeSet$
 - $RModels((N - C) \cup \neg C, UpdLits - C, \neg C, M, p)$
 4. /* Otherwise, determine whether N is a final result model. */
 - Otherwise, /* i.e. $Incons = \{\}$ */
 - Call $MinimalChange(N, M, p)$

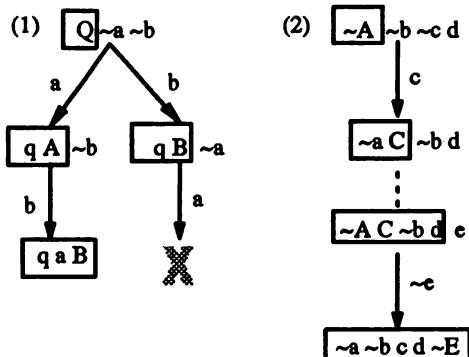
Figure 5: The Result Models Generation Algorithm (*RModels*)

Input: A candidate result model N , the initial model M , and a priority level p .
Output: Add N to $Result$ according to the minimal change semantics defined in PMA.
step:

1. If $p = StartPriority$
 Let $p' = \max(\text{priority levels in } PS)$
 else Let $p' = p$
2. For $i = p'$ downto p
/* Determine if N is farther away from M than the result models already found. */
If $PriDiff(r, M, i) \subset PriDiff(N, M, i)$, for some $r \in Result$
 Return /* not minimally changed; ignore N */
/* Otherwise, remove those models farther away from M than N . */
Let $Result = (Result - \{r \in Result \mid PriDiff(N, M, i) \subset PriDiff(r, M, i)\}) \cup \{N\}$

Figure 6: The Minimal Change Algorithm (*MinimalChange*)

right branch will be pruned out since it will not lead to any model with fewer changes than the one found in the left branch. There will be only one result model $\{a, b\}$. The second search tree corresponds to the case where $L = \{a, b, c, d, e, \neg a, \neg b, \neg c, \neg d, \neg e\}$, $M = \{a, \neg b, \neg c, d, e\}$, $I = \{\neg a \neg c, ac, \neg b \neg d, bd, \neg a \neg e, a \neg e, b \neg e, ab\}$ corresponding to the set of non-updatable formulas $\{(a \vee c) \wedge \neg(a \wedge c), (b \vee d) \wedge \neg(b \wedge d), e \leftrightarrow (a \vee b), \neg a \vee \neg b\}$, and $A = \{\neg a\}$. This is the same example used in Section 4 (Figure 1) except that now the predicate symbols of a, b, c , and d are given higher priority than that of e . As a result of prioritization, $\{\neg a, \neg b, c, d, \neg e\}$ is the only final result model.



The Minimal Change Algorithm

MinimalChange, as shown in Figure 6, implements the PMA's minimal change semantics. Given an initial model M and two candidate result models M_1 and M_2 , M_1 is considered closer to M than M_2 is (and therefore preferred over M_2) if $Diff(M_1, M) \subseteq Diff(M_2, M)$, where $Diff(L_1, L_2)$ returns the difference between sets of literals L_1 and L_2 . More precisely, $Diff(L_1, L_2) = \{l \mid l \text{ is a positive literal} \wedge l \in (L_1 - L_2) \cup (L_2 - L_1)\}$. Being invoked at each priority level, *MinimalChange* uses a function $PriDiff(L_1, L_2, p)$, which returns $Diff(L_1, L_2)$, re-

stricted to literals whose predicates have priority level p or higher.

The Model Generation Algorithm

We last present *MODELS*, which, given a finite set of literals A , finds a complete set of models for A . *MODELS* works by finding all literals that are entailed by A , then choosing one additional literal from the undecided ones (those in L but not entailed by A) and moving first it and then its negation to A , with recursive calls to *MODELS* for each new version of A . For example, let I be $\{\{a, \neg a\}, \{b, \neg b\}, \{\neg a, \neg b\}\}$, L be $\{a, \neg a, b, \neg b\}$, and A be $\{a\}$. Then *MODELS*(L, L, A) returns the set of models $\{a, \neg b\}, \{a, b\}$.

MODELS is a variant of *RModels*. The main difference between these two algorithms is that *MODELS* computes a complete set of models and for this reason, prioritization and the minimal change semantics are not applicable to *MODELS*.

7 Complexity Analysis

In this section, we analyze the complexity of the algorithms presented in Section 6. We focus our attention on the complexity of the recursive procedure *RModels* and its relationship to *Prioritize* and *MinimalChange*. Since *MODELS* is a variant of *RModels*, their functionality is the same and therefore the complexity of *MODELS* is not analyzed separately here. The performance of *INCON*, which is mainly involved in the pre-computation of minimal inconsistencies, will not affect the major activity of the system — finding result models — and therefore the complexity of *INCON* is also ignored.

Given a model of a theory and an update to the model, *RModels* searches for result models by calling itself recursively in a depth-first fashion. The conceptual

search tree implicit in this search process has the following properties:

- Each of the nodes, including the leaves, corresponds to an invocation of *RModels*.
- Each of the edges represents a flipping.
- Each of the leaf nodes represents an intermediate or final result model.

In English, *RModels* does the following:

1. Finds the subset E of inconsistencies where each inconsistency in this subset is a subset of the given candidate result model. (This subset of inconsistencies must all have been caused by the update or flipping applied to the predecessor of the candidate result model.)
2. Removes all non-updatable literals, those which are made true by the update or flippings, from the elements of E obtained in step 1.
3. Computes the set F of flippings from E obtained in step 2, and checks that each flipping in F is consistent with the non-updatable formulas of the theory.
4. Determines if a result model is final.

The complexity of each of these steps will be analyzed below. We begin by concentrating on the situation involving one priority level with one initial candidate result model. We will then extend our analysis to cover multiple levels of prioritization.

The bottom line is: let f be the cardinality of the largest flipping. Let h be the height of the conceptual search tree, and let w be the largest branch factor, the number of outgoing edges from a node, in the tree. Let I_d be the cardinality of the largest element in I , the set of minimal inconsistencies. Let I_l be the cardinality of the largest subset of inconsistencies where there is some literal common to all the inconsistencies in this subset. The worst case complexity for finding result models is

$$O(w^h \cdot f \cdot I_l \cdot I_d^{f+1})$$

and that for determining final result models according to the minimal change semantics defined in PMA is

$$O(w^{2h} \cdot f \cdot h).$$

Of most concern is the size of exponents h and f . Parameter h captures the ripple effects of an update and parameter f reflects the number of literal changes caused by an update. Assuming that the world is only slightly changed by an update, both h and f are small. There is a delicate interplay between h and I ; one can reduce h by including additional inconsistencies in I . These tradeoffs have been partially explored in [de Kleer 90]. Parameter I_d will at worst grow linearly

with the number of literals. Parameter I_l , which measures the number of interacting constraints affecting a particular literal, will at worst grow linearly with the set of minimal inconsistencies. Lastly, Parameter w measures the number of flippings at a node of the search tree; w is minimized by employing search pruning heuristics whenever possible, to avoid pursuing flippings that cannot lead to a final result model.

The details of our analysis follow:

The complexity of finding inconsistencies

The first step of *RModels* is to find all subsets s of the literals which make up the given candidate result model N such that s is itself an inconsistency. Since inconsistency s must have been caused by the update or flipping applied to the predecessor of N , s must contain some literals from the update or flipping. By using the index on the set of minimal inconsistencies, s can easily be found. Let f, I_l, I_d be defined as above. Then there are at most $f \cdot I_l$ inconsistencies in N . For each of these inconsistencies, we must determine if it is a subset of N . If a model is represented by a bit map and each literal by a bit address, the subset tests can be done in linear time, i.e., proportional to the number of literals to be tested. Since there are at most I_d literals in an inconsistency, the complexity of the first step in the worst case is

$$O(f \cdot I_l \cdot I_d).$$

The complexity of removing non-updatable literals

Next, we need to remove those literals that are not to be changed from the inconsistencies found in step 1. Let us call these literals non-updatable literals. These literals were made true in the given candidate result model by the given update or flippings. If the non-updatable literals are kept track of by a bit map, it will again take linear time to locate these literals in the inconsistencies. So, the complexity for this step is also

$$O(f \cdot I_l \cdot I_d).$$

The complexity of computing flippings

The third step involves computing flippings from the set of inconsistencies obtained in step 2 above and determining whether these flippings are consistent with the non-updatable formulas of the theory. Computing flippings is equivalent to computing cross-products. For instance, given a set of inconsistencies $\{\{a, b\}, \{c, d\}\}$, the corresponding set of flippings will be $\{\{\neg a, \neg c\}, \{\neg b, \neg c\}, \{\neg a, \neg d\}, \{\neg b, \neg d\}\}$. Unfortunately, there is no short-cut to computing cross-products. We choose to expand the inconsistencies in a depth-first manner, i.e. exhaust all possible combinations for a before dealing with b in the previous example. The minimality and consistency of a flipping is

checked along with the expansion process. A few data structures are used to help out the minimality and consistency checks: an *o-list*, which can be thought of as an array of lists, one for each of the literals appearing in the inconsistencies, is used to keep track of the occurrences of literals in the inconsistencies, which are assumed numbered; a bit vector, *visited*, is used to keep track of the direct or indirect involvement of inconsistencies; and the flipping being constructed is stored in a bit vector to ease consistency check.

It is necessary to set up the *o-list* before the expansion process takes place, and the cost for setting up the *o-list* is

$$O(f \cdot I_l \cdot I_d).$$

The expansion tree for computing flippings is characterized by the following parameters: its height is f ; its branch factor, i.e. the number of edges coming out of each node, is at most I_d ; and therefore the total number of edges in the tree is at most $O((I_d)^f)$. Each edge in the expansion tree represents the need for checking the minimality and consistency of the evolving flipping due to the introduction of a new literal into the flipping. To check for minimality, the entry in the *o-list* corresponding to the newly introduced literal is consulted and, if necessary, the bits in *visited* are marked appropriately. Since the length of an entry of an *o-list* is at most $f \cdot I_l$, the cost for the minimality check is $O(f \cdot I_l)$. The cost for checking whether the newly formed flipping is still consistent with the non-updatable formulas of the theory is $O(I_l \cdot I_d)$, corresponding to the need of scanning at most $I_l \cdot I_d$ number of literals. Hence, the total cost for each edge is $O(f \cdot I_l \cdot I_d)$ and the worst case complexity for this step is

$$O(f \cdot I_l \cdot I_d^{f+1}).$$

The complexity of finding result models

Now, we can evaluate the complexity of finding result models by *RModels* by combining the costs incurred in the three steps discussed above. Let w be the largest branch factor and h the height of the conceptual search tree. Then, the total number of nodes in the search tree is at most $O(w^h)$ and the total number of internal nodes is also at most $O(w^h)$.

The first step is executed for every invocation of *RModels*. The corresponding worst case cost for this step is

$$O(w^h \cdot f \cdot I_l \cdot I_d).$$

The second step is taken for each internal node of the search tree. The corresponding worst case cost for this step is also

$$O(w^h \cdot f \cdot I_l \cdot I_d).$$

The third step, which is also invoked in the internal nodes of the search tree, incurs a cost of

$$O(w^h \cdot f \cdot I_l \cdot I_d^{f+1}).$$

Hence, the overall worst case complexity of the three steps combined is

$$O(w^h \cdot f \cdot I_l \cdot I_d^{f+1}).$$

The complexity of finding final result models

The final step of *RModels* is to determine whether a result model found is final (relative to the result models found so far). This step is invoked only at the bottom level, the leaf nodes, of the conceptual search tree. For a particular leaf node in the search tree, *MinimalChange* is actually invoked to determine whether the corresponding result model is final. There are two steps in *MinimalChange*: first, *MinimalChange* checks to see if the result model has more changes than any result models accumulated so far; if the test is negative, the result model is final (again, relatively speaking). The second step checks and eliminates those result models which have more changes than the newly found final result model. Let n be the number of result models accumulated so far (but not including the newly found). The cost for testing any two models on changes is $O(f \cdot h)$ since there are at most $f \cdot h$ literals being changed in any particular model. Then, the cost of a particular invocation of *MinimalChange* will be

$$O(n \cdot f \cdot h).$$

Since there are at most w^h leaf nodes in the search tree, *MinimalChange* will be invoked at most $w^h - 1$ times. And since

$$\sum_{n=1}^{w^h-1} n = \frac{w^h \cdot (w^h - 1)}{2}$$

the worst case complexity for finding final result models will be

$$O(w^{2h} \cdot f \cdot h).$$

The complexity of computing result models with prioritization

Assuming that the number of priority levels is bounded by a constant, the inclusion of multiple priority levels does not change the computational complexity of Immortal's algorithms. We do expect prioritization, however, to have a major impact on average case performance as explained below.

Prioritization eliminates some result models by prohibiting changes on literals at higher priority levels. Consequently, w and f will both be smaller. The height, h , of a particular search tree at a particular priority level will not be greater than the height of the search tree without prioritization. As a result, the size of a particular search tree at a particular priority level will typically be smaller than the search tree without prioritization. There are more search trees in the case of prioritization because of iterations as compared to only one when there is no prioritization. But the reduction in height and branching factor in the search

Table 1: Run Time Performance of Immortal (Example 1)

Objs(m)	Ducts(n)	Run time (sec.)	Result models	$ I $	$ L $	I_d	I_l	I_{ref}
6	3	.07 - .1	1	140	56	7	10	28
8	4	.1 - .13	1	286	90	9	13	34
18	10	.17 - .27	1	2948	418	19	29	66
30	14	.42 - .52	1	10184	930	31	45	98
44	20	.87 - .97	1	30070	1890	45	65	138

At priority level 1: $w = f = h = 1$. At level 2: $w = f = 1; h = 2$.

Table 2: Run Time Performance of Immortal (Example 2)

Objs(m)	Ducts(n)	Run time (sec.)	Result models	$ I $	$ L $	I_d	I_l	I_{ref}
6	3	.07 - .1	3	140	56	7	10	121
8	4	.1 - .17	4	286	90	9	13	214
18	10	1.2 - 1.42	10	2948	418	19	29	1224
30	14	3.73 - 4.07	14	10184	930	31	45	2708
44	20	10.5 - 11.4	20	30070	1890	45	65	5626

At priority level 1: $f = h = 1; w = n$. At level 2: $w = h = 1; f = 2$.

trees will compensate for the constant factor growth in their number. In a nutshell, prioritization cuts down the computational effort for finding final result models.

8 Experimental Results

In this section, we show the use of two examples to measure the performance of Immortal, as written in KCL. The experiments were done on a Sun workstation.

The first example is adapted from [Ginsberg 87]. Very briefly, the examples involve the placement of m objects in $n + 1$ locations in a room. n of the locations are ducts, and the remaining location is the floor. The non-updatable formulas include “type” axioms: objects can only be placed on locations, and locations cannot be put on anything. The world is also governed by the following non-updatable formulas: (1) an object cannot be at two different locations at the same time; (2) no two different objects are at the same location unless it is the floor; (3) every object must be located somewhere; (4) a duct is blocked iff there is some object on it; and (5) the room will be stuffy when all ducts are blocked. Predicates are divided into two priority levels: object locations, and room stuffiness.

In Example 1, the updatable formulas state that all ducts are blocked with objects and the update is to move the object from duct₁ to the floor. In Example 2, the updatable formulas state that all objects are on the floor except one which is on duct₁ and the update is to make duct₁ clear. Table 1 and 2 show the run time, as measured by KCL’s “time” function, for performing the updates of Examples 1 and 2. We consider the results very reasonable; it took less than a second

for computing a model in the first example and less than 0.6 second in the second case. The results also confirm that when the ramifications of an update are shallow, as we expect them to be for applications suitable for minimal change semantics, the run time of the system grows only linearly with the size of the problem, as measured by the size of the set I of minimal inconsistencies. Also shown in Table 1 and 2 is the growth in the total number of references to elements of I (I_{ref}).

The second example is selected from [Forbus 89]. This example deals with changes of a physical system using qualitative simulation augmented with actions. Qualitative simulations are of particular interest to us in that the underlying model (states and transitions) of qualitative simulations fits well with the belief revision model (models and revisions). Also the minimal change belief revision semantics are useful for reasoning about the effects of actions or dynamic processes in the physical world. The example involves a small pot which can be placed either under a faucet, on a stove, or on the table. The physics about the phase changes of water contained in the pot is encoded as dynamic state transitions with pre- and post-conditions. These dynamic transitions are assumed to be pre-compiled and directly available to us. *Envisioning* in Forbus’ system involves the computation of all legal physical states, followed by the computation of all legal transitions among these states, where the transitions can be made by actions or triggered by dynamic processes. Our envisioning is different from Forbus’: we compute states (models) in stages. For some given state, we find the set of actions and dynamic transitions applicable to it and update the state with each of these actions and transitions. The new states are the union of the

result states obtained from each of these updates less the ones already found before.

80 states and 392 transitions resulted from our envisioning process, which started with the initial state/model where the pot is on the table and both the faucet and stove are not turned on. Involved in the experiment were 20 literals, 19 inconsistencies, 8 possible actions, and 9 possible dynamic state transitions. The envisioning process took 12.42 seconds to complete; the computation of all but the initial state took 7.74 seconds and the rest (4.68 seconds) was spent on combining duplicate states.

9 Conclusion

Our experience with implementing Immortal shows that model-based belief revision is a promising knowledge representation and reasoning tool for dealing with reasonably large problems. We are in the process of extending Immortal to handle changes in universe composition and function interpretation, so that we can apply Immortal to problems from more complex domains.

Acknowledgement

This work has been supported by NASA under grant NAG 1-613 and by NSF under grant IRI 89 58582. The authors would like to thank Ken Forbus and Alan Frisch for helpful comments on an earlier draft of this paper. Our appreciation also goes to Michael Barbenn, Scott Bennett, Jon Gratch, Peter Haddawy, and David Page for helpful discussions and support.

References

- [Abiteboul 85] S. Abiteboul and G. Grahne, "Update Semantics for Incomplete Databases," *Proceedings of the Conference on Very Large Data Bases*, Stockholm, August 1985.
- [Alchourrón 85] C. E. Alchourrón, P. Gärdenfors, and D. Makinson, "On the Logic of Theory Change: Partial Meet Contraction and Revision Functions," *The Journal of Symbolic Logic*, 50:2, pp. 510-530, June 1985.
- [Dalal 88] M. Dalal, "Investigations into a Theory of Knowledge Base Revision: Preliminary Report," *Proceedings of AAAI-88*, Minneapolis, pp. 475-479, August 1988.
- [de Kleer 86] J. de Kleer, "An Assumption-Based TMS," *Artificial Intelligence* 28:2, pp. 127-162, 1986.
- [de Kleer 90] J. de Kleer, "Exploiting Locality in a TMS," *Proceedings of AAAI-90*, Boston, pp. 264-271, July 1990.
- [Forbus 89] K. D. Forbus, "Introducing Actions into Qualitative Simulation," *Proceedings of the Eleventh Int'l Joint Conference on Artificial Intelligence*, Detroit, pp. 1273-1278, August 1989.
- [Gärdenfors 88a] P. Gärdenfors and D. Makinson, "Revisions of Knowledge Systems Using Epistemic Entrenchment," *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, Asilomar, pp. 83-95, March 1988.
- [Gärdenfors 88b] P. Gärdenfors, *Knowledge in Flux: Modeling the Dynamics of Epistemic States*, Bradford Books, MIT Press, 1988.
- [Ginsberg 87] M. L. Ginsberg and D. E. Smith, "Reasoning About Action I: A Possible Worlds Approach," in M. L. Ginsberg (ed.), *Readings in Nonmonotonic Reasoning*, Los Altos, CA, Morgan Kaufmann, 1987.
- [Hegner 87] S. Hegner, "Specification and Implementation of Programs for Updating Incomplete Information Databases," *Proceedings of the Sixth Symposium on Principles of Database Systems*, San Diego, pp. 146-158, March 1987.
- [Katsuno 89] H. Katsuno and A. O. Mendelzon, "A Unified View of Propositional Knowledge Base Updates," *Proceedings of the Int'l Joint Conference on Artificial Intelligence*, Detroit, August 1989.
- [Mitchell 89] M. Mitchell, *The Anatomy of a Framework for Resolution-based Automated Proof Procedure Systems*, M.S. Thesis, Computer Science Department, University of Illinois, 1989.
- [Rao 89] A. S. Rao and N. Y. Foo, "Minimal Change and Maximal Coherence: A Basis for Belief Revision and Reasoning about Actions," *Proceedings of the International Joint Conference on Artificial Intelligence*, Detroit, August 1989.
- [Reiter 87] R. Reiter, "A Theory of Diagnosis from First Principles," *Artificial Intelligence*, April 1987.
- [Sandewall 89] E. Sandewall, "Filter Preferential Entailment for the Logic of Action in Almost Continuous Worlds," *Proc. of the Int'l Joint Conference on Artificial Intelligence*, Detroit, August 1989.
- [Satoh 88] K. Satoh, "Nonmonotonic Reasoning by Minimal Belief Revision," *Proceedings of the Int'l Conference on Fifth Generation Computer Systems 1988*, ICOT, pp. 455-462, December 1988.
- [Weber 86] A. Weber, "Updating Propositional Formulas," *Proceedings of the First International Expert Database Systems Conference*, Charleston SC, pp. 487-500, April 1986.
- [Winslett 88] M. Winslett, "Reasoning About Action Using a Possible Models Approach," *Proceedings of the Seventh National Conference on Artificial Intelligence*, Minneapolis, pp. 89-93, August 1988.

Clustering Temporal Intervals To Generate Reference Hierarchies

William S. Davis
 Amoco Production Company
 Research Center
 4502 East 41st St., P.O. Box 3385
 Tulsa, OK 74102
 bdavis@trc.amoco.com

James R. Carnes
 Boeing Computer Services
 Artificial Intelligence Center
 P.O. Box 240002, MS JY-58
 Huntsville, AL 35824
 ray@hsvaic.boeing.com

Abstract

Many reasoning systems which much represent and manipulate knowledge about time-related events employ the interval-based temporal logic initially developed by James Allen. While this representation facilitates a description of the possible temporal relationships among various world events, the computation required to manipulate these relationships and maintain consistency is very expensive. The primary expense in this computation is the time to propagate new relationships to obtain a consistent closure throughout existing ones. Work has been done to lessen this propagation expense using limited sets of temporal relationships, limited transitivity tables, and hierarchical abstraction. This paper outlines a new method for automatic hierarchical abstraction of temporal relationships by arranging temporal intervals into clusters and generating temporal reference intervals to hierarchically preside over these clusters. The paper describes techniques for clustering and maintaining the reference hierarchy, as well as methods for posting (recording and propagating) constraints across hierarchical boundaries.

1 INTRODUCTION

Interval-based temporal logic [Allen83a, Allen85] has become a very popular paradigm for representing and maintaining knowledge about time-related events. Many problem-solving systems which must address "real world" situations encounter the need to reason about events which are temporally interrelated. While automated planners

may be the most ubiquitous of such systems [Allen83b, Hogge87a], other applications include scheduling [Meng90], qualitative reasoning [Hogge87b] and spatial reasoning [Mukerjee90]. The interval logic maintains knowledge of time periods (intervals) that are bounded by time points, with relations between two intervals being represented as a disjoint set of the possible logical (temporal) relationships which exist between them. Most work in the interval logic recognizes that these possible relationships consist of seven mutually-exclusive temporal operators (before, meets, overlaps, starts, during, finishes, and equals) and their inverses — thirteen in all ("equals" is its own inverse). Therefore a relationship from one interval to another is the set of all possibilities of temporal arrangement between the two intervals that are consistent with the other temporal relationships in the system. Whenever a new assertion relating intervals is presented, the system propagates any effects of this assertion by attempting to reduce the possibilities (operators) existing between these intervals and any related intervals. Any relationships that are affected by this propagation are themselves propagated until the system reaches the logical closure of the initial assertion.

Completeness in propagating relationships among intervals in this fashion has been shown to be an NP-hard problem [Vilain86]. However, a commonly used incomplete algorithm which suffices for many reasoning situations is described in [Allen83a] and has been shown to require $O(n^3)$ time. Nonetheless, even with polynomial time complexity, this propagation becomes prohibitive as the number of intervals in the system grows. Several approaches have been taken to manage this complexity. One such approach is to trade the expressiveness of the full interval algebra for a simpler representation involving *point relations* rather than interval relations [Vilain86, Allen89]. However, this point-based approach is inadequate for representing natural events and actions that occur in most "real world" situations. Another approach has been to determine a minimal set of the thirteen temporal operators while maintaining the same

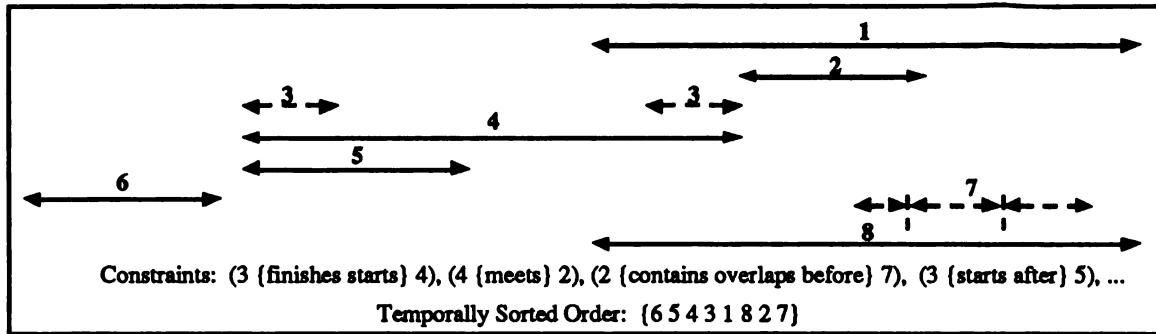


Figure 1: Example of sorting temporal intervals

expressiveness [Zhu87, Matuszek88], thus reducing the size of the transitivity table used in constraint propagation. While this approach somewhat reduces propagation time by using a smaller table, propagation still occurs across each pair of intervals and works to produce a completely connected network of intervals (just as in the original constraint propagation). A third approach to managing complexity focuses on this completely connected network of intervals. Allen originally proposed to reduce complexity by abstracting sets of intervals into *reference intervals*. Reference intervals are hierarchically superior intervals which initially hold a *containment relationship* (started-by, contains, finished-by, equals) to their subintervals. Any proposition which holds over the reference interval naturally holds over all its subintervals, and a relationship between two intervals *A* and *B*, not directly related, may be derived by transitivity over the temporal links "up" from interval *A* to a common reference and back "down" to interval *B*.

1.1 REFERENCE HIERARCHIES

Constructing a reference hierarchy over a set of intervals prior to constraint propagation, however, is quite labor-intensive. Furthermore, manual maintenance (*i.e.*, by a user of a temporal reasoning system) of this hierarchy can be completely impractical for domains which require a dynamic hierarchy. The need for a predefined reference structure is circumvented with automatic construction of reference intervals. A good method for automatic referencing has been developed [Koomen89] which is based on temporal containment. In effect, this system promotes existing intervals to the status of reference intervals by virtue of their containment links to other intervals. Any direct links that can be derived by transitivity through the reference hierarchy are eliminated, thus reducing the number of (redundant) connections through which constraints must be propagated.

This approach is strictly limited, however, in that the system allows the direct (*i.e.*, non-hierarchical) relationships between intervals to contain only single operators. That is, the relationships between intervals

under the same reference must be fully determined, rather than the usual set of temporal possibilities. This is not always practical for temporal reasoners whose purpose is to derive these relationships, such as in the applications listed earlier. Another limitation of this approach is that using existing intervals as reference intervals may reflect a false hierarchy in some domains. [Allen90] gives a simple example of simultaneous events required in opening a locked door, where a spring lock must be held open with one hand, while the door is pulled open with the other hand. Imposing a constraint such as *HoldLockOpen* (*contains*) *PullDoor* might imply that *PullDoor* is hierarchically inferior to *HoldLockOpen*, when in fact both would more appropriately be subintervals to some previously unnamed *OpenDoor* operation.

This paper describes further work on a method for automatic generation of temporal reference intervals which uses machine-generated references to hierarchically preside over temporally-clustered intervals. The reference hierarchy is maintained by imposing a threshold on the number of subintervals allowed per reference parent. The next section details this approach, followed by a discussion of constraint propagation within the reference hierarchy and its effect on dynamic hierarchy adjustment.

2 GENERATING REFERENCE INTERVALS

2.1 CLUSTERING

Assuming for this section that the only constraints posted are those between "comparable" intervals (*i.e.*, within the same reference), the amount of time spent in propagating constraints is directly proportional to the number of intervals within that reference. Hence, controlling the number of intervals per reference interval is very desirable in controlling propagation complexity. The method used for constructing the reference hierarchy is therefore based on the ability to determine how to divide a group of *N* intervals into *p* partitions each of cardinality $\leq k$, where *k* is a pre-specified constant dictating the hierarchy breadth, and $p = \text{ceiling}(N/k)$.

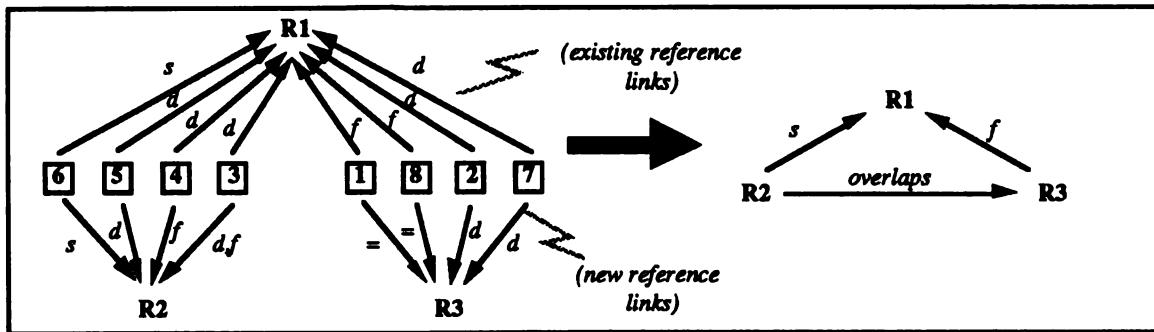


Figure 2: Deriving sibling constraints among newly generated reference intervals R2 and R3

A system has been implemented which dynamically builds a reference hierarchy from simple temporal constraints posted between interval pairs. Intervals being constrained which are not already known to the system are added to the current reference.¹ Each reference interval possesses a machine-generated label and is distinct from any intervals named externally; this allows intervals which are related using containment operators (*starts*, *during*, *finishes*, *equal*) to remain as subintervals to a hierarchically superior reference interval. When a given reference interval becomes oversubscribed (*i.e.*, more subintervals than allowed by the threshold constant), it is divided into two or more new reference intervals, with each new reference interval inheriting less than k of the former reference's subintervals. To illustrate this, we will describe the simple case of dividing the oversubscribed reference into two partitions.

To partition subintervals of a network within the same reference interval (hereafter known as a "local network"), an attempt is made to group the intervals into *temporal clusters*. The current system sorts the intervals according to their increasing starting endpoints, with ties in start times broken by increasing order on the finishing endpoints. The difficulty in obtaining such a sort is that the system deals with multiple possibilities in temporal relationships, not just singleton relationships as in the Koomen reasoner. Hence, we say the intervals are sorted into their *likely* relative timeline order. This likelihood is determined by the following heuristic:

interval i is "less than" interval j if there is more evidence in the local network relationships to suggest that i starts before j than there is to suggest that j starts before i.

Figure 1 shows an example of such a sort. This method tends to organize the intervals into moderately sequential groups of more interrelated intervals, thus clustering them into relative temporal localities. A loose analogy can be

made between this approach and that of the graph topology technique which uses articulation points to determine connected components of a graph. This approach is especially effective in domains whose intervals often have a relatively sequential order.

2.2 RESTRUCTURING

Once sorted, the system virtually partitions the intervals into an "earlier" half and a "later" half. "Virtually" here means that the two clusters are considered separate networks, but constraint propagation still considers them connected for the time being. In the example of Figure 1, the clusters would be {6 5 4 3} {1 8 2 7}. New reference intervals are created for these halves and (*starts*, *during*, *finishes*, *equals*) constraints are initially established as *reference links* between members of the halves and their respective new reference interval. The system computes whether any of the intervals in each half respectively start, finish, or equal their new reference interval, and posts such constraints. Note that propagating these reference links throughout both halves automatically derives a relationship between the new reference intervals, and from the new reference intervals to the former reference interval. This is illustrated in Figure 2 by extending the previous example. The intervals, currently under the reference R1, are divided into two halves, {6 5 4 3} and {1 8 2 7}. New reference intervals R2 and R3 are placed over the halves, and the relationships R2 (*overlaps*) R3, R2 (*starts*) R1 and R3 (*finishes*) R1 are results (among others) of propagating the reference links from {6 5 4 2 1 8 2 7} to R2 and R3.

The following algorithm outlines this generation of new reference intervals. This algorithm calls upon a few self-explanatory functions to sort the intervals, partition them into clusters, create new reference intervals for each cluster, and post the appropriate temporal constraints to derive relationships among the newly generated and former reference intervals. The function *ExtractRelations(I, N)* returns all relations in the temporal network N involving intervals only from interval set I.

¹ *current reference* is derived from the context of the constraint

```

To RestructureHierarchy (ref_interval)
;;subnet is the temporal network under ref_interval
subnet ← ref_interval.sub-network
subintervals ← TemporallySort(subnet)
P ← Partition(subintervals, k)
;; where k is max intervals per reference
for each partition Pi do
begin
  Ri ← CreateNewReferenceIntervalFor(Pi)
  ;;use only temporal relations involving intervals in Pi
  Pi.local_net ← ExtractRelations(Pi.intervals, subnet)
  for each interval j in Pi do
    begin
      ref_link ← {starts during finishes equal} |
                  {starts} | {finishes} | {equal}
      ;;depending on j's temporal position in Pi.local_net
      Post(j ref_link Ri) to subnet
      RecordReferenceLink(j ref_link Ri)
    end
  end
end

```

Once propagation is complete, the halves are "physically" partitioned by disallowing future propagation to carry across any relationships involving intervals from both halves. Constraints derived between intervals in separate partitions are recorded for future retrieval of relationships, but not propagation. The hierarchy restructuring is performed by adding the new reference intervals to the local network which contains the former reference interval. Any appropriate start/finish/equal information between the new reference intervals and the former reference interval is posted in the former's local network, allowing the new reference intervals to inherit (through transitivity) relationships to the former reference's sibling intervals. The final step is then to delete the former reference interval from its local network, determine whether its network has now become oversubscribed by the addition of the new reference intervals, and if so, repeat the above procedure on successively higher levels in the reference hierarchy until either no more adjustment is necessary or the root of the hierarchy is reached. The algorithm for this is as follows (using the variable context of the previous algorithm).

```

for each relationship (x temp_ops_set y) in subnet,
  where x and y are in different partitions do
begin
  RecordCrossLink(x temp_ops_set y)
end
super_ref ← ref_interval.reference-interval
sibling_net ← super_ref.sub-network
P_names ← ExtractPartitionNames(P)
;;post relations involving only newly created
;;reference intervals and/or ref_interval to the local
;;network containing ref_interval
N ← ExtractRelations(P_names + ref_interval, subnet)

```

```

for each relationship (x temp_ops_set y) in N do
begin
  Post(x temp_ops_set y) to sibling_net
end
for each new reference interval Ri do
begin
  AddSubInterval(super_ref, Ri)
end
DeleteSubInterval(super_ref, ref_interval)
if Cardinality(super_ref.sub-intervals) > k then
  RestructureHierarchy(super_ref)

```

3 POSTING CONSTRAINTS IN THE HIERARCHY

The reference hierarchy generally limits constraint propagation to those intervals within a given reference interval (the local network). The hierarchy is effective in abstracting the temporal relationships, but information from temporal assertions can be lost when abstracted through the reference structure. That is, constraints which are deduced through hierarchical transitivity between some pairs of cross-hierarchy (non-local) intervals can be weaker (*i.e.*, contain more temporal possibilities) than their original, pre-restructuring constraint. For this reason constraints between subinterval partitions that are dropped from the hierarchy when restructuring are remembered as *cross-links* and associated with an individual interval. Hence information that had been previously asserted or derived between two intervals will not be lost if restructuring separates them, even though constraint propagation will not follow these cross-links.

Posting constraints to two intervals within the same reference is done as in typical "flat" networks. The challenge for the hierarchical system is to post relationships between intervals in different subtrees of the reference hierarchy, without reverting the hierarchy back to a flat network. Two methods have been identified for asserting these cross-hierarchy constraints. The first is called the *Nearest Common Ancestor* (NCA) method. Rather than combining the subtrees containing the intervals in the cross-constraint (thereby flattening that portion of the hierarchy), NCA uses this constraint to modify the sibling relationship which already exists under their "nearest common ancestor." This is the lowest (*i.e.*, farthest from the hierarchy root) reference interval which is a hierarchical ancestor of both intervals being constrained. NCA builds a separate temporal network composed of all intervals and relationships along the path from each constrained interval up to the common ancestor. The cross-hierarchy constraint is then posted to this separate network, and the relationship between the two subintervals of the common ancestor which are ancestors of the originally constrained intervals is extracted. This extracted relationship is then propagated among the subinterval relationships of the nearest common ancestor.

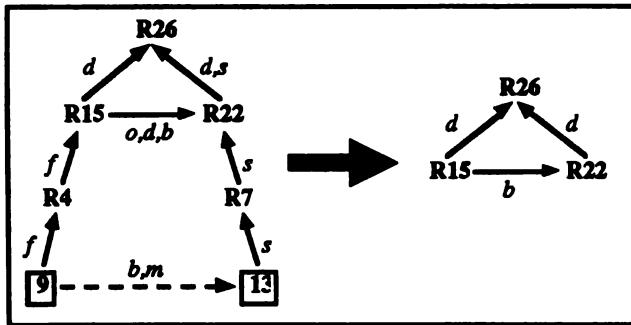


Figure 3: NCA Constraint Posting

The original constraint is also stored with one of the originally constrained intervals as a cross-link, just as in hierarchy restructuring.

Figure 3 illustrates NCA posting of the constraint $\{before\} 9 \{before\} 13$. Given that R26 is the nearest common ancestor for intervals 9 and 13, and that R15 and R22 are their respective ancestors directly under R26, a temporary network is composed of the intervals {9 R4 R15 13 R7 R22 R26}. The relationships in this network consist of their hierarchy relationships plus the sibling relationship ($R15 \{overlaps\} R22$). Posting the cross-hierarchy constraint to this temporary network would constrain relationships in the hierarchy as follows:

```
RecordCrossLink (9 {before} 13)
Post (R15 {before} R22) to R26.sub-net
RecordReferenceLink (R22 {during} R26)
```

The second method for posting cross-hierarchy constraints, called the Collapse method, involves collapsing the reference structure between the intervals being constrained and their nearest common ancestor. Existing reference links, sibling links within each reference, and any cross-links stored with the intervals themselves are used to reduce this portion of the hierarchy to a flat network. The cross-constraint is then posted and propagated within this combined network. The restructuring methods outlined above are used to rebuild the hierarchy, with the root of the rebuilt hierarchy replacing the previous nearest common ancestor. This collapse-and-rebuild method is more complete than the NCA method: constraint propagation is more thorough, and intervals have the chance to be redistributed under different reference intervals during repartitioning. However, the overhead in collapse and rebuild operations may eliminate any propagation savings gained over the flat representation. Several factors, including domain characteristics and the order in which assertions are made, make it difficult to predict whether this method is useful. Work to fully determine the effectiveness of this method is in progress; the current system employs the NCA method.

4 CONCLUSIONS

4.1 ANALYSIS

The primary result from using this technique is that propagation delays in deriving closures of temporal assertions are effectively controlled with a threshold on allowable subintervals per reference. This is borne out by experiments which have been performed with a system using the techniques of temporal clustering, hierarchy restructuring, and NCA constraint posting to deduce relationships among crew operations for manned spacecraft activity. To illustrate performance differences between the flat and hierarchical representations, constraints were generated which posted random sets of temporal operators between random intervals in the network. Table 1 summarizes a subset of these experiments using a hierarchy breadth of 10 intervals per reference to structure from 10 to 50-interval networks (posting random constraints numbering roughly 1.5 times the number of connections in the flat network to assure reasonable constraining of the relationships). As expected, the time for propagation remained relatively constant when hierarchy restructuring was used, which accounts for the overwhelming savings in total time for posting all constraints.

This methodology has benefits over other known approaches in that it allows disjunction in inter-reference sibling links, and thresholds on hierarchy breadth can be adjusted to suit machine resources, domain concerns, and needs for consistency and completeness. However, there are some limitations which are introduced by such benefits. The main sacrifice for the marked reduction in complexity is a more incomplete consistency check of temporal constraints since propagation is restricted within reference intervals. A second effect of this restriction is that some of the cross-hierarchy relationships are weaker than if they had been determined in the flat network. These relationships contain several more temporal operators than their flat network counterparts. Additionally, some relationships in the local networks (i.e., under the same reference interval in the hierarchy) may also be slightly weaker due to limited propagation. However, these limitations may be suitable for some domains which exhibit characteristics which localize relationships (e.g. planning for moderately sequential activity). Since the constraints are clustered into temporally local space, it is possible that the weaker cross-links would be seldom used. Storing cross-hierarchy relationships (from assertions and restructuring) in the cross-links between lower-level intervals also lessens these limitations.

Table 1: Performance comparison between hierarchy and flat representations

# of nodes	# of random constraints	relative time units to post all constraints	
		hierarchy	flat network
10	70	1	1
20	300	5	14
30	650	19	85
40	1150	37	420
50	1850	52	768

4.2 Future Work

While empirical results look promising, large sets of domain data are required for better analysis. Experiments using randomly generated constraints tend to simulate worst case scenarios, perhaps even unrealistically so. It might be assumed that as constraints are added, and the hierarchy dynamically adjusted, increasingly more constraints will focus on the high degree of interrelationships in the local networks, particularly in the Collapse method. Random constraints, however, do just the opposite by statistically favoring cross-constraints as the number of intervals in the network far exceed the hierarchy breadth threshold. If the acquisition of substantial sets of domain constraints becomes prohibitive, this randomness deficiency may be overcome with a biasing scheme in generating the constraints.

Another effort toward improving analysis is deriving better means of characterizing "information loss" among NCA and Collapse methods. While such characterization is often dependent on the domain, work could be done to identify trends in the various stages of abstraction which contribute to weaker temporal relationships. More study is required of the Collapse method to determine its effectiveness, including provisions for dynamic hierarchy balancing. Time orders for algorithmic complexity must be derived for both NCA and Collapse methods; it is currently thought that the complexity follows the pattern of a "union-find" type of algorithm.

Another area for investigation is a more intelligent partitioning scheme that makes better use of temporal information to determine where to divide a sorted set of intervals, rather than just choosing equal partitions. Ultimately, these techniques in automated hierarchy generation could be used in an attempt to discover interval clusters, possibly deriving interval groups for applications such as learning new planning operators.

Acknowledgements

This work was performed under independent research and development by The Boeing Company. The authors wish to thank the conference reviewers for their helpful comments, as well as George Williams and Jeff Jackson for their insightful reviews of the paper.

References

- [Allen83a] Allen, J.F., "Maintaining Knowledge about Temporal Intervals", *Communications of the ACM*, vol. 26, pp.832-843.
- [Allen83b] Allen, J.F. and Koomen, J.A., "Planning Using a Temporal World Model", *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 741-747.
- [Allen85] Allen, J.F. and Hayes, P.J., "A Common-Sense Theory of Action and Time", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 528-531.
- [Allen89] Allen, J.F. and Hayes, P.J., "Moments and Points in an Interval-Based Temporal Logic", *Computational Intelligence*, vol. 5, pp. 225-238.
- [Allen90] Allen J.F., "Temporal Reasoning and Planning" in Reasoning about Plans, draft manuscript.
- [Hogge87a] Hogge, J.C., "TPLAN: A Temporal Interval-Based Planner with Novel Extensions", University of Illinois Dept. of Computer Science Technical Report UIUCDCS-R-87-1367, September 1987.
- [Hogge87b] Hogge, J.C., "Compiling Plan Operators from Domains Expressed in Qualitative Process Theory", *Proceedings AAAI-87*, pp. 229-233.
- [Koomen89] Koomen, J.A., "Localizing Temporal Constraint Propagation", *Proceedings Principles of Knowledge Representation and Reasoning*, pp. 198-202.
- [Matuszek88] Matuszek, D., et. al., "Endpoint Relations on Temporal Intervals", Paoli Research Center Technical Report PRC-LBS-8810, June 1988.
- [Meng90] Meng, A.C., and Raja, B.A., "Logos-TCS: An Expert System for Operations Management based on Temporal Constraint Satisfaction", *Proceedings*

- Sixth IEEE Conference on Artificial Intelligence Applications*, pp. 215-221.
- [Mukerjee90] Mukerjee, A. and Joe, G., "A Qualitative Model for Space", *Proceedings AAAI-90*, pp. 721-727.
- [Vilain86] Vilain, M. and Kautz, H., "Constraint Propagation Algorithms for Temporal Reasoning", *Proceedings AAAI-86*, pp. 377-382.
- [Zhu87] Zhu, M., Loh, N. and Siy, P., "Towards the Minimum Set of Primitive Relations in Temporal Logic", *Information Processing Letters*, vol. 26, pp. 121-126.

Default Logic Revisited

James P. Delgrande
W. Ken Jackson
School of Computing Science
Simon Fraser University
Burnaby, B.C.
Canada V5A 1S6

Abstract

In Reiter's default logic, classical first-order logic is augmented with a set of domain-specific rules of inference for drawing conclusions concerning default properties of individuals. These rules are of the form $\alpha : \beta/\gamma$. The intended reading is "if α can be proven from what's known, and β is consistent, then conclude γ ". Unfortunately, the approach does not always provide the results that one might intuitively expect: in some cases "reasonable" default conclusions are not forthcoming; in others, "unreasonable" conclusions are. We argue that the problems arise because, first, the requirement of proving the antecedent α is too strong and, second, the notion of consistency given for the justification β is too weak. In the first case we argue that one needs only defaults of the form $: \beta/\gamma$, where $\beta \supset \gamma$. In the second case we modify the manner in which consistency is used in the definition of a default extension. Combining the approaches leads to a more general formulation of default theories which arguably better conforms to intuitions regarding consistency-based default reasoning. We show too that one can now reason *about* a set of defaults and can determine, for example, if a particular default in a set is redundant. Lastly we show how two of these variants are related to the Theorist system.

1 INTRODUCTION

[Reiter 80] describes an extension to first-order logic for "jumping to conclusions" concerning default prototypical properties of individuals. In this approach, which we will refer to as *default logic*, domain-specific rules of inference or *defaults* are added to classical propositional or first-order logic to capture patterns of inference of the form "in the absence of information to

the contrary, conclude that ...". The form of a default is $\frac{\alpha(\vec{x}) : \beta(\vec{x})}{\gamma(\vec{x})}$. Informally it may be interpreted as: "If, for some set of instances \vec{c} , $\alpha(\vec{c})$ is provable from what is known and $\beta(\vec{c})$ is consistent, then conclude by default that $\gamma(\vec{c})$ ". For example, "birds fly" may be represented by the rule $\frac{\text{Bird}(x) : \text{Fly}(x)}{\text{Fly}(x)}$. This is interpreted as "if something can be inferred to be a bird, and if that thing can be consistently assumed to fly, then infer that that thing flies". Hence the semantics of the approach rests on notions of provability and consistency with respect to sets of beliefs. A central concern is how an agent might consistently extend its set of beliefs, given a set of default statements.

However there are situations where default logic appears to be too weak, in that "reasonable" default conclusions are not forthcoming; also there are situations where it appears too strong, in that unwanted default conclusions are obtained. In this paper we present three modifications to default logic that address these issues and, in so doing, extend the range and applicability of default logic while simplifying its machinery.

For the first set of problems, where desirable conclusions are not forthcoming, we argue that the requirement that the antecedent be provable is too strong. This requirement is weakened by transforming the defaults so that there is no prerequisite condition. From the transformed rules one can draw default conclusions that could not be made before. This transformation also leads to a simplified, iterative specification of a default extension, or set of beliefs that may reasonably be held. Moreover, it is now possible to reason about a set of defaults, and to determine logical relations among defaults. This variant is referred to as P-default logic (for *prerequisite-free* default logic).

For the second problem, we strengthen the notion of consistency used in forming an extension. This modification, referred to as J-default logic (for *justification* default logic), arguably captures a more appropriate notion of consistency-based default reasoning.

The two modifications together, PJ-default logic, are shown to fully address the aforementioned problems and so provide an intuitive, general variant of default logic. In these approaches the semantic characterisation given in [Etherington 87] is amended to provide a corresponding characterisation for these modifications. Lastly, we show that P-default logic and PJ-default logic are very closely related to the Theorist system of [Poole 88]; hence the results reported in the paper provide a link among these systems.

The next section provides a brief summary of default logic. Section 3 develops P-default logic for normal defaults first, then for semi-normal defaults. Section 4 describes J-default logic and subsequently PJ-default logic. Section 5 discusses the relation with Theorist. The final section provides a general discussion.

2 DEFAULT LOGIC

The approach of default logic is to add rules of inference to first-order logic to capture patterns of inference of the form “in the absence of information to the contrary assume ...”. The general form¹ of a default is $\frac{\alpha(\vec{x}) : \beta(\vec{x})}{\gamma(\vec{x})}$. This rule may be informally interpreted as: “If, for some set of instances \vec{c} , $\alpha(\vec{c})$ is provable from what is known and $\beta(\vec{c})$ is consistent, then conclude by default that $\gamma(\vec{c})$.” $\alpha(\vec{x})$ is called the *prerequisite*; $\beta(\vec{x})$ is the *justification*; and $\gamma(\vec{x})$ is the *conclusion*. A *normal* default is one where the justification and conclusion are the same; a *semi-normal* default is one where the justification implies the conclusion. Almost all “naturally occurring” defaults are normal [Etherington 88]; semi-normal defaults are required for “interacting” defaults [Reiter and Crisculo 81]. A *default theory* is a pair (W, D) where W is a set of first-order (or propositional) formulae and D is a set of defaults. A *closed* default theory is one where none of the formulae in the defaults contains a free variable. For simplicity we deal only with closed default theories; open theories are easily dealt with, as described in [Reiter 80].

A set of defaults induces one or more *extensions* of the formulae in W . An extension can be viewed as an acceptable set of beliefs that one may hold about the world W , suitably augmented by the defaults D . We have:

Definition 2.1 ([Reiter 80]) If S is a set of first-order sentences, and $\Gamma(S)$ is the smallest set satisfying the three properties:

1. $W \subset \Gamma(S)$.

¹Actually, this is not wholly general, since there may be more than one consistency condition in a default. However, the results reported here extend easily to multiple consistency conditions.

2. $\Gamma(S) \subseteq Th(\Gamma(S))$. (That is, $\Gamma(S)$ is closed under first-order consequence.)

3. If $\frac{\alpha : \beta}{\gamma} \in D$ and $\alpha \in \Gamma(S)$ and $\neg\beta \notin \Gamma(S)$ then $\gamma \in \Gamma(S)$.

then E is an extension of the default theory (D, W) iff $\Gamma(E) = E$.

That is, E is a fixed point of Γ . We also have:

Theorem 2.1 ([Reiter 80]) Let (W, D) be a closed semi-normal default theory. Define:

$$\begin{aligned} E_0 &= W \\ E_{i+1} &= Th(E_i) \cup \{\gamma \mid \alpha : \beta / \gamma \in D \\ &\quad \text{and } \alpha \in E_i \text{ and } \neg\beta \notin E\}, \text{ for } i \geq 0 \end{aligned}$$

Then E is an extension for (W, D) iff $E = \bigcup_{i=0}^{\infty} E_i$.²

Hence for example, if we had that Quakers are typically pacifists while Republicans typically are not (i.e.

$\frac{Q(x) : P(x)}{P(x)}$ and $\frac{R(x) : \neg P(x)}{\neg P(x)}$) then, if we were given that $Q(sue)$ and $R(sue)$ were true, there would be two extensions, one in which $P(sue)$ was true and one in which $\neg P(sue)$ was true. If we also had the rules that $\frac{Q(x) : G(x)}{G(x)}$ and $\frac{A(x) : E(x)}{E(x)}$ (say, that Quakers are typically generous and adults are typically employed) then $G(sue)$ would be true in both extensions, and $E(sue)$ would be true in neither.

In the following sections we modify the above construction, given in theorem 2.1, to define new notions of an extension; the fixed point definition will then in all cases follow as a property of each type of an extension.

Our informal interpretation of a default rule is that it is, or should behave like, a defeasible conditional. That is, it should ideally retain the properties of classical material implication, unless these properties are explicitly blocked. Under this reading, the construction of an extension can be viewed as ensuring that modus ponens applies to instances of a rule unless it is explicitly blocked via the justification. Transitivity between default rules, or between a rule and a material implication, holds unless otherwise blocked. [Reiter and Crisculo 81] in large part shows how such transitivities may be blocked using semi-normal defaults.

However, as mentioned, this formulation appears weaker than we might otherwise like. First, we cannot reason by cases, and so given

$$\frac{Q(x) : P(x)}{P(x)} \text{ and } \frac{V(x) : P(x)}{P(x)}$$

²Note the occurrence of E in the definition of E_{i+1} .

together with $Q(\text{leslie}) \vee V(\text{leslie})$, we cannot conclude $P(\text{leslie})$, because neither $Q(\text{leslie})$ nor $V(\text{leslie})$ alone is provable. In addition, we cannot reason using the contrapositive of a default. For example, if we have the rule that birds typically fly, $\frac{\text{Bird}(x) : \text{Fly}(x)}{\text{Fly}(x)}$,

and that $\neg\text{Fly}(\text{opus})$ is true, then it seems reasonable to conclude by default that $\neg\text{Bird}(\text{opus})$ is true. However, such a conclusion is not forthcoming in default logic unless one explicitly gives the rule $\frac{\neg\text{Fly}(x) : \neg\text{Bird}(x)}{\neg\text{Bird}(x)}$. It seems preferable though

that the contrapositive of a default hold, by default, unless it is explicitly blocked. Lastly, in default logic one cannot reason about defaults. Thus while in classical logic, if we know that $\alpha \supset \gamma$ and $\beta \supset \gamma$ are true, then $\alpha \vee \beta \supset \gamma$ must also be true. However in default logic, there is no explicit connection between $\frac{\alpha : \gamma}{\gamma}$,

$$\frac{\beta : \gamma}{\gamma} \text{ and } \frac{\alpha \vee \beta : \gamma}{\gamma}.$$

These considerations can be summarised by noting that it would seem to be a guiding principle of defeasible reasoning that a default should behave like the material conditional³ except that it is defeasible. Hence, for example, modus ponens should apply except when it is known to lead to an inconsistency. This indeed is the case for default logic. However, other principles of conditionals, such as reasoning by cases or by modus tollens, do not hold "by default". So one goal here is to be able to incorporate these other forms of default reasoning. Of course, what this also means is that there will be cases where we may not want to be able to reason via modus tollens (say), and so will need to be able to also block such use of the default rules.

As mentioned too, in other cases default logic is stronger than one might want. For example, as [Poole 88] points out, the defaults $\frac{: (A \wedge B)}{A}$ and

$$\frac{: (C \wedge D)}{C}$$
 together with $\neg B \vee \neg D$ yield a single extension where both A and C are true. This conflicts with one's intuitions, since one of B or D must be false and so one of the default conditions cannot hold, and so both default conclusions should not jointly hold. The difficulty here (effectively) is that in definition 2.1 (and in theorem 2.1) justifications need only be individually consistent with an extension.

There are of course numerous other approaches to defeasible reasoning, including approaches that avoid these pitfalls. This however does not mean that default logic should perhaps be abandoned in favour

of one of these alternatives. First, default logic is one of the best known approaches to defeasible reasoning, and has been thoroughly studied [Reiter 80, Etherington 88, Konolige 87, Kautz and Selman 89], and used in a wide variety of differing applications, e.g. [Etherington and Reiter 83, Mercer 88, Perrault 87, Reiter 87]. Second, other approaches to defeasible reasoning arguably address somewhat differing problems from default logic. For example [Geffner 88] and [Neufeld and Poole 88] deal with probabilistic connections; [Poole 88] deals more with implementation strategies for defeasible reasoning; [McCarthy 80] and others address minimising a predicate's extension; and [Delgrande 88] addresses a non-consistency-based approach to defeasible reasoning using a conditional logic. [Krauss et al 90] presents a number of families of Gentzen-style nonmonotonic consequence relations, which are related to Shoham's notion of a model preference relation [Shoham 87]. Of other consistency-based approaches, [McDermott and Doyle 80] and [McDermott 82] have some reasonably unintuitive properties, and [Moore 85], while formally appealing, is not obviously implementable. Hence it is useful and relevant to re-examine default logic to see if the aforementioned difficulties cannot be avoided.

In the following sections we will deal only with normal and semi-normal defaults. We note in closing that these are the only forms that need to be considered, in that general defaults give no increase in generality over semi-normal defaults. Consider where we have a default theory (W, D) , with $\frac{\alpha : \beta}{\gamma}$ in D . This default is appropriate in the specification of an extension (in theorem 2.1) precisely when $\frac{\alpha : \beta \wedge \gamma}{\gamma}$ is. To see this, consider where we have an extension E , obtained from D and world knowledge W . There are three cases:

1. If $\neg\beta$ is provable from E then neither $\alpha : \beta / \gamma$ nor $\alpha : \beta \wedge \gamma / \gamma$ can be used in the specification of E .
2. If both $\neg\beta$ and $\neg\gamma$ are not provable from E then from the definition of E , $\alpha : \beta / \gamma$ and $\alpha : \beta \wedge \gamma / \gamma$ can be used interchangeably in the specification of E .
3. If $\neg\beta$ is not provable from E but $\neg\gamma$ is provable from E , then by the definition of E , $\alpha : \beta / \gamma$ cannot have been used in the specification of E : assuming to the contrary, we would have that $\gamma \in E$ by application of the default; by assumption $\neg\gamma \in E$; since $\gamma \wedge \neg\gamma \supset \neg\beta$ from FOL and E is closed under classical consequence, we have $\neg\beta \in E$, contradicting the original assumption. Secondly, since $\neg\gamma$ is provable from E , the semi-normal default $\alpha : \beta \wedge \gamma / \gamma$ also cannot be used in the specification of E .

³This of course is highly informal and is intended as a rule of thumb only. In particular we note that defeasible reasoning is realised here by means of rules of inference, whereas the material conditional is a sentential operator.

3 P-DEFAULT LOGIC

We have argued that in default logic we should be able to reason by cases or reason using the contrapositive of a conditional, unless such properties are explicitly blocked. The rationale is that a default conditional should (by default) retain properties of the classical conditional, unless explicitly blocked. The emphasis then shifts to the conditional itself, rather than a rule involving a prerequisite and justification for a conclusion. For simplicity, we deal first with normal defaults. The next subsection extends these results to semi-normal defaults.

3.1 NORMAL DEFAULTS

A normal default $\frac{\alpha : \beta}{\beta}$ is read as "typically, if α then β " or, more precisely, as "if α must be true and β is consistent, then β ". The reading we propose for "typically, if α then β " instead is "if it is consistent that $\alpha \supset \beta$ then conclude that $\alpha \supset \beta$ ". Informally, in "typically if α then β " we take "typically" to govern the conditional. Thus in this reading "if α then β " should be concluded (when consistent) regardless of the provability of α . Hence, we replace a default

$\frac{\alpha : \beta}{\beta}$ with the prerequisite-free form $\frac{\alpha \supset \beta}{\alpha \supset \beta}$.

In this variant, called P-default logic, a conditional, when consistent, is asserted. This in turn means that we can reason with this conditional. "Birds fly", expressed propositionally by $\frac{\text{Birds} \supset F}{\text{Birds} \supset F}$, is read "if it is consistent that a bird flies then conclude that the bird flies".

Consider what this means for reasoning by cases. If we have that $D = \left\{ \frac{Q : P}{P}, \frac{V : P}{P} \right\}$ together with $W = \{Q \vee V\}$ then since neither Q nor V are provable, in default logic neither rule is applicable and we can conclude nothing about P . However intuitively this seems to be a little too strong. We know that one of Q or V must be true, and that P follows by default from either Q or V . So it seems that we should be able to conclude by default that P . Again, it appears that we want to be able to conclude (by default) the conditionals as a whole, and then subsequently be able to reason with these conditionals.

That is, the translation to prerequisite-free defaults is

$$\frac{Q : P}{P} \mapsto \frac{: Q \supset P}{Q \supset P} \text{ and } \frac{V : P}{P} \mapsto \frac{: V \supset P}{V \supset P}.$$

With $W = \{Q \vee V\}$ both defaults are applicable. Thus there is a single extension containing $Q \vee V$, $Q \supset P$, $V \supset P$, and so P .

Now, it may be countered that the same result is obtained by rewriting the defaults so that the prerequisites contain the various cases in a disjunction.

For example if the above defaults are rewritten as $\frac{Q \vee V : P}{P}$ then P follows by default. Unfortunately,

in default logic there is no way to determine if this default expresses the same thing as the original defaults. As we show at the end of this section, not only are prerequisite-free defaults simpler, but one can formally demonstrate that if the truth value of P is not known

$$\frac{: Q \supset P}{Q \supset P} \text{ and } \frac{: V \supset P}{V \supset P}$$

have the same effect as

$$\frac{: (Q \vee V) \supset P}{(Q \vee V) \supset P},$$

since $(Q \vee V) \supset P$ is consistent if and only if $Q \supset P$ is consistent and $V \supset P$ is consistent.

Consider next reasoning with the contrapositive. If we have "typically if α then β ", then by our reading of "typically" as applying to the conditional as a whole, we should also have "typically if $\neg\beta$ then $\neg\alpha$ ", unless we have reason to not accept the contrapositive. Again, we can accomplish this if the conditional itself is concluded whenever consistent. That is, for the defaults

$$\frac{A : B}{B} \text{ and } \frac{: A \supset B}{A \supset B}.$$

and for $W = \{\neg B\}$ nothing would be concluded given the first default; for the translated default there is a single extension that contains $\neg B$, $A \supset B$ and thus $\neg A$. There will however be times when we will want to block the contrapositive; this issue is addressed in the next subsection.

First we describe more formally the advantages of prerequisite-free defaults. First, in the theorem below, we show that the translation to prerequisite-free defaults results in a more general system, in that one can conclude more in this new system. Second, the definition of an extension is simplified by taking into account the form of prerequisite-free defaults. Finally, a logic for reasoning about prerequisite-free defaults is described.

Theorem 3.1 Let (W, D) be a normal default theory and let (W, D') be the theory where $\alpha : \beta / \beta \in D$ iff $:(\alpha \supset \beta) / (\alpha \supset \beta) \in D'$. If E is an extension of (W, D) then there is an extension of (W, D') , E' , such that $E \subseteq E'$.

We may also get more extensions from (W, D') . For example the default theories,

$$\begin{aligned} &\left(\{A\}, \left\{ \frac{A : B}{B}, \frac{B : \neg A}{\neg A} \right\} \right) \text{ and} \\ &\left(\{A\}, \left\{ \frac{: A \supset B}{A \supset B}, \frac{: B \supset \neg A}{B \supset \neg A} \right\} \right) \end{aligned}$$

have 1 extension and 2 extensions respectively.

Using prerequisite-free defaults, the procedure for constructing an extension can be simplified. A default, $\alpha : \beta / \beta$ is applicable in the specification of E_{i+1} only if $\alpha \in E_i$. For prerequisite-free defaults there is no prerequisite, so this test can be dropped. More importantly, the occurrence of E in the definition can be replaced by E_i . This is because for any prerequisite-free normal default β / β , if $\neg\beta \in E$ then it must be the case that for some i that $\neg\beta \in E_j$ for every $j \geq i$. With these changes, an extension for P-default logic is defined as follows.

Definition 3.1 Let (W, D) be a closed normal prerequisite-free default theory. Define

$$\begin{aligned} E_0 &= Th(W) \\ E_{i+1} &= Th(E_i \cup \{\beta\}) \text{ where } : \beta / \beta \in D \\ &\quad \text{and } \neg\beta \notin E_i, \text{ for } i \geq 0 \end{aligned}$$

Then E is a P-extension for (W, D) iff $E = \bigcup_{i=0}^{\infty} E_i$.

This definition is simpler and we would expect some speedup in a procedure for constructing P-extensions, even though the overall complexity of the approach remains unchanged. That is, the revised definition yields two heuristic advantages: first we have no prerequisite condition to prove, and, second, if for default β / β , β is inconsistent with E_i then β will be inconsistent with every E_j , $j \geq i$. Any procedure for constructing a P-extension can ignore all such “inapplicable” defaults. This is not the case for a standard default $\alpha : \beta / \beta$ since it may be inapplicable due to the fact that α is not provable in E_i , although α may well be provable in some E_{i+j} .

A more important consequence is that we can now reason about normal defaults. So, given a set of prerequisite-free normal defaults, it is now possible to ask whether a default $\alpha : \alpha / \alpha \in D$ is “subsumed” by other defaults in the set, in that one can ask whether the P-extensions of (W, D) and $(W, D - \alpha)$ coincide. Thus for example, if C is not known to be false then, in the specification of an extension, the defaults $\frac{: A \supset C}{A \supset C}$ and $\frac{: B \supset C}{B \supset C}$ have precisely the same effect as $\frac{: A \vee B \supset C}{A \vee B \supset C}$. Also, for example, any time that $\frac{: A \wedge B}{A \wedge B}$ may be used in forming a P-extension, so may $\frac{: A}{A}$ and $\frac{: B}{B}$. However the converse of this last relation does not necessarily hold: $\frac{: A}{A}$ and $\frac{: \neg A}{\neg A}$ may individually be applied in forming a P-extension, but $\frac{: (A \wedge \neg A)}{(A \wedge \neg A)}$ certainly cannot.

These notions can be made more precise as follows:

A default α / α is *applicable*, written $applic(\alpha)$, with respect to a set of formulae W if $\{\alpha\} \cup W$ is consistent. Given a knowledge base W (or a set of formulae E_i at some point in the definition of an extension), the set of applicable defaults is the subset of D that may be applied to W (or to E_i). We obtain:

Theorem 3.2

1. $applic(\alpha \wedge \beta) \supset applic(\alpha) \wedge applic(\beta)$.
2. $applic(\alpha \vee \beta) \equiv applic(\alpha) \vee applic(\beta)$.
3. $applic(\alpha) \equiv applic(\neg\neg\alpha)$.
4. $\alpha \supset applic(\alpha)$.
5. If $\alpha \supset \beta$ then $applic(\alpha) \supset applic(\beta)$.
6. If $\neg\alpha$ then $\neg applic(\alpha)$.

If we define $\Box\alpha$ as $\neg applic(\neg\alpha)$ then the resulting logic is just KT [Chellas 80] (also called T [Hughes and Cresswell 68]). It is usually axiomatised using the formulae:

$$\Box\alpha \supset \alpha$$

and

$$\Box(\alpha \supset \beta) \supset (\Box\alpha \supset \Box\beta)$$

and the rule:

$$\text{If } \alpha \text{ then infer } \Box\alpha.$$

Items 2, 5, and 6 in the theorem are easily shown to be adequate to derive these relations. The notion of truth is provided using a possible-worlds semantics, where the accessibility relation between worlds is reflexive only. Since “applicable” as used above is synonymous with “is consistent with”, and since it seems to make no sense here to talk about iterated occurrences of the modality *applicable*, these relations *exactly* specify the system that we have. In this case, $\Box\alpha$ would have the informal reading “is provable”.

As an example, if we were given that

$$D = \left\{ \frac{: A \supset C}{A \supset C}, \frac{: B \supset C}{B \supset C} \right\}$$

then we can prove that these defaults can be employed at some step in the specification of an extension whenever $\frac{: A \vee B \supset C}{A \vee B \supset C}$ can, unless $\neg C$ is provable. That is, from theorem 3.2 we can prove that

$$\begin{aligned} \vdash &(applic(A \supset C) \wedge applic(B \supset C)) \\ &\supset (applic(A \vee B \supset C) \vee \neg applic(C)) \end{aligned}$$

In the case where we have $W = \{A \vee B, \neg C\}$, there are two extensions given D above, one in which A and $\neg B$ are true and another in which $\neg A$ and B are true. However the condition $A \vee B \supset C$ is inconsistent with W , and so nothing additional can be concluded given only the default $\frac{: A \vee B \supset C}{A \vee B \supset C}$.

This means that we can now regard default reasoning as being composed of two distinct and disjoint parts. First, we have the notion of using the defaults to construct a (P-)extension, or acceptable set of beliefs that hold by default. Now in addition we have a means of *reasoning about* a set of formulae and a set of defaults to determine, for example, whether or not a particular default is “subsumed” by others, and so can be discarded. This is specified in theorem 3.2. This division into two parts is basically the approach taken in [Delgrande 88] for default reasoning.

3.2 SEMI-NORMAL DEFAULTS

While most “naturally occurring” defaults are normal, there are cases where effectively we want one default to “take precedence” over others. Thus we would want the default that states penguins normally don’t fly to take precedence over that that states birds normally fly. Such interacting defaults must often be expressed using semi-normal defaults. This section describes the need for semi-normal defaults, and shows how prerequisite-free semi-normal defaults can be used to address problems with interacting defaults.

Consider the statements “university students are typically adults”, “adults are typically employed”, and “students are typically not employed”. These statements may be represented with the defaults,

$$\frac{: S \supset A}{S \supset A}, \frac{: A \supset E}{A \supset E}, \text{ and } \frac{: S \supset \neg E}{S \supset \neg E}.$$

Three extensions result from $W = \{S\}$, corresponding to the following:

1. student \Rightarrow adult, student \Rightarrow \neg employed
2. student \Rightarrow adult \Rightarrow employed
3. student \Rightarrow \neg employed \Rightarrow \neg adult

Note that the third extension uses the contrapositive of the second default. We would like to block the transitivities implicit in the second and third extensions and just use the more specific defaults that “students are typically adults” and “students are typically not employed”.

The issue of interacting defaults was first addressed in [Reiter and Criscuolo 81]. There the authors showed how semi-normal defaults, with prerequisites, could be used to block unwanted transitivities.⁴ In the preceding example, “adults are normally employed” would, in isolation, be written as $A : E/E$. The transitivity of inferring that a student is employed, using the second default, would be prevented by rewriting this default

⁴They also show how one may get by with normal defaults if additional assumptions can be made. However, the difficulty is that we may not always be willing to make such assumptions.

as $\frac{A : E \wedge \neg S}{E}$. Our approach is an extension of this to prerequisite-free semi-normal defaults. An additional concern here is that we must also block contrapositives (such as the contrapositive of the second default that produced the third extension above).

For a semi-normal default, there are two possible translations:

$$\frac{\alpha : \beta \wedge \gamma}{\beta} \leftrightarrow \frac{: (\alpha \supset \beta) \wedge \gamma}{(\alpha \supset \beta)}$$

and

$$\frac{\alpha : \beta \wedge \gamma}{\beta} \leftrightarrow \frac{: \alpha \supset (\beta \wedge \gamma)}{\alpha \supset \beta}.$$

The first possibility carries γ as a “global” consistency condition, while the second uses γ as a “local” consistency condition, under the assumption of α . The two alternatives are closely related since $\frac{: \alpha \supset (\beta \wedge \gamma)}{\alpha \supset \beta}$

is applicable whenever $\frac{: (\alpha \supset \beta) \wedge \gamma}{(\alpha \supset \beta)}$ is. We have found the first alternative to be preferable, because it blocks the contrapositive of $\alpha \supset \beta$ when γ is inconsistent; blocking this contrapositive is more complicated using the second alternative. Hence only the first alternative will be considered from now on.

In general then, if a prerequisite-free default $\frac{: \alpha}{\alpha}$ should not be applicable under condition γ then the default is written as $\frac{: \alpha \wedge \neg \gamma}{\alpha}$. The default “adults are typically employed” is not applicable for students so we may write

$$\frac{: S \supset A}{S \supset A}, \frac{: (A \supset E) \wedge \neg S}{(A \supset E)}, \text{ and } \frac{: S \supset \neg E}{S \supset \neg E}.$$

From $W = \{S\}$ we get a single extension that includes S , A , and $\neg E$. Blocking the contrapositive of a conditional, $\frac{: \beta \supset \delta}{\beta \supset \delta}$, is a special case of this where γ is $\neg \delta$. For example, the contrapositive of the default, “A’s are typically B’s” is blocked by writing the rule as $\frac{: (A \supset B) \wedge B}{A \supset B}$. If we are given $\neg B$ then clearly the default is inapplicable and so we cannot conclude $\neg A$.

The following theorem shows that the above translation gives a more general system.

Theorem 3.3 Let (W, D) be a semi-normal default theory and let (W, D') be the theory where $\alpha : (\beta \wedge \gamma) / \beta \in D$ iff $: (\alpha \supset \beta) \wedge \gamma / (\alpha \supset \beta) \in D'$. If E is an extension of (W, D) then there is an extension of (W, D') , E' , such that $E \subseteq E'$.

Unfortunately, with this extension to semi-normal defaults, we do not retain the benefits obtained with

normal defaults. For forming an extension, for E_{i+1} , we would now have to make reference to the final extension E instead of just the previous step E_i . Also we cannot reason about the set of applicable defaults. However, in the next section, wherein J-default logic is introduced, we show that in the amalgamation of the two approaches, PJ-default logic, we regain these benefits.

4 J- AND PJ-DEFAULT LOGIC

In some cases default logic produces conclusions that, intuitively, are stronger than one wants. For example, if we have the defaults $\frac{:\ A \wedge B}{A}$ and $\frac{:\ C \wedge D}{C}$ together with the knowledge that $\neg B \vee \neg D$ then we can conclude by default that both A and C . This clearly clashes with one's intuitions. Informally, since one of B or D is false, one of the default conditions (we don't know which) cannot hold, and so both default conclusions should not jointly hold. The difficulty is that both in the original formulation of an extension, and in the formulations given to this point, if we know that $\neg B \vee \neg D$ then the possibility (or consistency) of B justifies the application of the first default, while the separate possibility of D justifies the second. This suggests that we require the set of justifications used in the specification of an extension be consistent, rather than each individual justification.

As a second example, if we have the defaults $\frac{:\ A \wedge B}{A}$ and $\frac{:\ \neg B}{\neg B}$ then in the original approach we obtain only one extension, in which B is false⁵. But again this seems unintuitive: if we know nothing else originally, then there seems to be no good reason that the first default not be "applicable". The argument might run: "Initially I know nothing at all; hence $A \wedge B$ is consistent, and I can conclude that A . However $\neg B$ is inconsistent with my original assumption, and so I cannot apply the second default." Similar reasoning beginning with the second default yields a second extension.

The definition for specifying a J-extension is a modification of theorem 2.1; the resultant system is called J-default logic. In this definition, a J-extension is composed of two sets of formulae: E_J , a set of justifications and E_T , the subset of E_J assumed to be true. Although this complicates the definition of an extension, it also means that rules and extensions are now represented uniformly, in that both consist of a consistency condition along with conclusions based on the consistency conditions.

Definition 4.1 Let (W, D) be a closed semi-normal

⁵Recall that in theorem 2.1 the consistency condition is with respect to an extension E

default theory. Define

$$\begin{aligned} E_0 &= (E_{J0}, E_{T0}) \\ &= (\text{Th}(W), \text{Th}(W)) \\ E_{i+1} &= (E_{Ji+1}, E_{Ti+1}) \\ &= (\text{Th}(E_J \cup \{\beta \wedge \gamma\}), \text{Th}(E_{Ti} \cup \{\beta\})) \\ &\quad \text{where } i \geq 0, \\ &\quad \alpha : (\beta \wedge \gamma) / \beta \in D, \\ &\quad \alpha \in E_{Ti}, \\ &\quad \neg(\beta \wedge \gamma) \notin E_{Ji}. \end{aligned}$$

Then E is a J-extension for (W, D) iff

$$E = (E_J, E_T) = (\bigcup_{i=0}^{\infty} E_{Ji}, \bigcup_{i=0}^{\infty} E_{Ti}).$$

The set E_J in a J-extension is used to accumulate the justifications from the applied defaults. Note that no reference is made in the definition of E_{i+1} to the extension E itself. For the example at the start of this section, we now obtain two J-extensions, one in which A is true and the value of C is unspecified, and another in which C is true and the value of A is unspecified. In the first J-extension, E_J contains the justification $A \wedge B$ from the first default. In the second J-extension, E_J contains the justification $C \wedge D$ from the second default. In the second example we obtain two extensions, one in which A is true and another in which $\neg B$ is true.

[Etherington 88] provides a semantic characterisation of extensions in default logic that is easily amended for J-extensions. This modification requires changing the notion of preference between models so that for a set of defaults D , the union of the justifications in the defaults holds in one of the preferred models. The soundness and completeness results of [Etherington 88] are also easily amended to provide similar results for J-default logic. We also obtain:

Theorem 4.1 Let (W, D) be a semi-normal default theory. If E is any extension of (W, D) obtained using theorem 2.1 then there is a J-extension $E' = (E'_J, E'_T)$ of (W, D) , obtained using definition 4.1, such that $E'_T \subseteq E$.

That is, J-default logic is weaker than standard default logic.

Finally we combine the modifications of the previous sections with those of the present to obtain PJ-default logic:

Definition 4.2 Let (W, D) be a closed prerequisite-free semi-normal default theory. Define

$$\begin{aligned} E_0 &= (E_{J0}, E_{T0}) \\ &= (\text{Th}(W), \text{Th}(W)) \\ E_{i+1} &= (E_{Ji+1}, E_{Ti+1}) \end{aligned}$$

$$\begin{aligned}
 &= (Th(E_{Ji} \cup \{\beta \wedge \gamma\}), Th(E_{Ti} \cup \{\beta\})) \\
 \text{where } i &\geq 0, \\
 &:(\beta \wedge \gamma)/\beta \in D, \\
 &\neg(\beta \wedge \gamma) \notin E_{Ji}.
 \end{aligned}$$

Then E is a PJ-extension for (W, D) iff

$$E = (E_J, E_T) = (\bigcup_{i=0}^{\infty} E_{Ji}, \bigcup_{i=0}^{\infty} E_{Ti}).$$

Note that this definition leads easily and immediately to a procedure for forming a PJ-extension: if E_0 is initialised to (W, W) , then E_{i+1} can be set to $(E_{Ji} \cup \{\beta \wedge \gamma\}, E_{Ti} \cup \{\beta\})$ where $:(\beta \wedge \gamma)/\beta \in D$ and $E_{Ji} \not\models \neg(\beta \wedge \gamma)$. A formula α now follows by default if it is provable from some E_{Ti} . The procedure is not generally computable since it appeals to nonprovability, $\not\models$. It is computable however in some restricted cases, notably in inheritance hierarchies. Lastly, it is substantially simpler than the procedure given in [Etherington and Reiter 83].

For PJ-default logic there are no prerequisites and the definition of E_{i+1} depends on E_i and not on E . Thus we obtain the benefits of the previous modifications: we can reason by cases and with the contrapositive (unless explicitly blocked); transitive defaults can be blocked using semi-normal defaults if necessary; and one can again reason about a set of defaults. On the other hand, the system isn't so strong that undesirable results are forthcoming. More pragmatically, the various situations explored in [Reiter and Crisculo 81] hold here, as do the applications developed in [Etherington and Reiter 83], [Mercer 88], [Perrault 87], and [Reiter 87].

5 RELATION TO THEORIST

Prerequisite-free default logic is closely related to the Theorist system [Poole 88]: P-default logic corresponds exactly to basic Theorist while PJ-default logic corresponds to Theorist with constraints.

A basic Theorist system consists of a set of closed formulae, F (the facts), and a set of formulae, Δ (the hypotheses). A scenario is a set $F \cup S$ where S is a set of ground instances of Δ such that $F \cup S$ is consistent. A Theorist extension is the deductive closure of a maximal scenario. For default reasoning, Δ contains conditionals similar to prerequisite-free defaults. For example, the following Theorist system corresponds to the Quaker-Republican-Pacifist example,

$$\begin{aligned}
 F &= \{Q, R\} \\
 \Delta &= \{Q \supset P, R \supset \neg P\}
 \end{aligned}$$

The two maximal scenarios are $F \cup S_1$ and $F \cup S_2$ where $S_1 = \{Q \supset P\}$ and $S_2 = \{R \supset \neg P\}$. Poole shows that a formula $w \in \Delta$ is a syntactic variant of the normal default $:w/w$. In P-default logic all defaults are

normal hence P-default logic corresponds exactly to Theorist. The contribution of P-default logic is the translation scheme from defaults to prerequisite-free defaults that tells us how to translate defaults to Theorist and theorem 3.1 that says we don't lose anything in this translation. The system developed in section 3 for reasoning about defaults could also be applied to reasoning about hypotheses in Theorist.

The problems of interacting defaults (blocking unwanted transitivities and contrapositives) also occurs in Theorist. The Theorist solution is to "name" defaults and to explicitly state when a default is not applicable. The default "typically, if A then E " might be given the name N_{AE} . Δ is restricted to contain only the names of defaults and F is augmented to include formulae such as $N_{AE} \supset (A \supset E)$. The names are useful to explicitly state when a default is not applicable. Recall that the above default is not applicable for students and this could be stated by including $S \supset \neg N_{AE}$ in F . However, this might introduce an unwanted side effect in that whenever N_{AE} is included in a scenario then $\neg S$ must follow from that scenario by using the contrapositive of $S \supset \neg N_{AE}$. "Constraints" were added to Theorist to state when a default is not applicable without any side effects.

A Theorist system with constraints consists of sets F and Δ as before plus a set C (the constraints) of closed formulae. A scenario in Theorist with constraints is a set $F \cup S \cup C$ where S is a set of ground instances of Δ such that $F \cup S \cup C$ is consistent. If C contains $S \supset \neg N_{AE}$ then the default N_{AE} cannot be applied when S is known and there are no side effects.

We used semi-normal defaults for exactly the same purpose: to state without side effects when a default is not applicable. [Poole 88] describes a method for translating Reiter's semi-normal defaults to Theorist and the method applies equally well to prerequisite-free defaults. The semi-normal default $\frac{:(A \supset E) \wedge \neg S}{(A \supset E)}$ corresponds to the Theorist fragment,

$$\begin{aligned}
 N_{AE} \supset (A \supset E) &\in F \\
 N_{AE} &\in \Delta \\
 S \supset \neg N_{AE} &\in C
 \end{aligned}$$

More generally, a default theory (W, D) can be translated into a corresponding Theorist system as follows:

$$\begin{aligned}
 F &= W \cup \{N_{\beta\gamma} \supset \beta \mid : \beta \wedge \gamma / \beta \in D\} \\
 \Delta &= \{N_{\beta\gamma} \mid : \beta \wedge \gamma / \beta \in D\} \\
 C &= \{\neg \gamma \supset \neg N_{\beta\gamma} \mid : \beta \wedge \gamma / \beta \in D\}
 \end{aligned}$$

where $N_{\beta\gamma}$ are the new unique names for the defaults. As shown in the next theorem, there is a close relationship between PJ-extensions and Theorist extensions.

Theorem 5.1 Let $E = (E_J, E_T)$ be a PJ-extension of (W, D) . If (F, Δ, C) is the Theorist system constructed from (W, D) by the above translation then there is a maximal scenario $F \cup S$ such that $E_T \subset Th(F \cup S)$.

It is easy to construct S from E by choosing the elements of Δ that correspond to the defaults applied to construct E . E_T is a strict subset because $\text{Th}(F \cup S)$ contains the names of the applied defaults. One might think that the names are not important but this is not the case. A simple counterexample is,

$$\begin{aligned} W &= \{\neg B \wedge A\} \\ D &= \left\{ \frac{: A \wedge B}{A}, \frac{: E \wedge F}{E} \right\} \end{aligned}$$

which, if names are not used, translates to

$$\begin{aligned} F &= \{\neg B \wedge A\} \\ \Delta &= \{A, E\} \\ C &= \{\neg B \supset \neg A, \neg F \supset \neg E\} \end{aligned}$$

In PJ-default logic there is a single extension where E_T contains A , $\neg B$ and E . In Theorist there are no possible scenarios since $F \cup C$ is inconsistent. (Incidentally, Theorem 5.1 in [Poole 88] asserts that in Theorist without constraints “naming” defaults has no effects on the possible scenarios. However, the above example shows that this theorem does not extend to Theorist with constraints.) The difference occurs because Theorist requires the entire set of constraints to be consistent with the facts while in PJ-default logic only the justifications used in building an extension need be consistent.

This close relationship between PJ-default logic and Theorist with constraints has benefits for both: we can reason about defaults in Theorist using the system developed in section 3 and the implementation techniques developed for Theorist could be used to implement PJ-default logic.

6 CONCLUSIONS

This paper has developed variants to default logic that address situations where the original system appears either too weak, in that “reasonable” default conclusions are not forthcoming; or too strong, in that some unwanted default conclusions can be drawn. In the first case, where some useful conclusions cannot be drawn, we suggest that the problem is solved by weakening the requirement that the prerequisite be provable. If a default rule is intended to capture conditionals of the form “typically if α then β ”, then our reading of this is that “typically” applies to the conditional as a whole. Thus the conditional is a component of the justification, as well as constituting the conclusion. In this modification, if the conditional can be concluded then one can reason using the conditional, without the need to prove the antecedent true. Thus one obtains contrapositive defeasible inferences, as well as reasoning by cases. Interacting defaults are dealt with by incorporating semi-normal defaults where necessary, in the manner described in [Reiter and Crisculo 81]. Semi-normal defaults are

also used to block other possibly-unwanted inferences, such as those involving contrapositives.

The second case, where unwanted conclusions may be drawn, is resolved by strengthening the notion of consistency in the definition of an extension. This involves ensuring that the set of justifications from the applied defaults is consistent with what is known (rather than each justification individually). Etherington’s semantics can also be appropriately modified to provide a semantic characterisation of such extensions.

The combination of these changes satisfactorily addresses the aforementioned difficulties and the approach gives rise to an intuitive and reasonable set of default inferences. That is, the system appears to be strengthened and weakened in just the right manner. That the suggested changes are indeed reasonable is justified first by appealing to the intuitions motivating these changes and secondly by observing that all “reasonable” inferences that we encountered in the literature still go through. We have also shown that the resulting system is closely related to the Theorist system of [Poole 88] and thus provide a link between Theorist and default logic.

From a technical standpoint, a number of benefits are obtained. First, the formulation appears conceptually cleaner and simpler. The modifications result in a uniformity of notation, in that both rules and extensions in a default theory are composed of two parts: a justification and a conclusion, where the justification implies the conclusion. The definition of an extension is somewhat simplified and, in particular, the specification of the set E_{i+1} depends now on E_i and not on the final extension E . While the overall complexity remains the same for default inferencing, there are now no prerequisites in the defaults to be proven. In addition, in the construction of an extension, if any default is inapplicable, it is always inapplicable. Hence we obtain a (heuristic) computational advantage. Thirdly, we can now reason about the defaults. Thus for example we can deductively determine that given the knowledge that $\neg B \vee \neg D$, the defaults $\frac{: A \wedge B}{A}$ and $\frac{: C \wedge D}{C}$ cannot be jointly applied.

Acknowledgments

This research was supported in part by the Natural Science and Engineering Research Council of Canada grant A0884. The first author is a member of the Institute for Robotics and Intelligent Systems (IRIS) and acknowledges the support of the Networks of Centres of Excellence Program of the Government of Canada, and the participation of PRECARN Associates Inc. The second author wishes to acknowledge support from a Natural Science and Engineering Research Council of Canada postgraduate scholarship.

References

- [Chellas 80] B.F. Chellas. *Modal Logic*. Cambridge University Press, 1980.
- [Delgrande 88] J.P. Delgrande. An approach to default reasoning based on a first-order conditional logic: Revised report. *Artificial Intelligence*, 36(1):63–90, 1988.
- [Etherington and Reiter 83] D.W. Etherington and R. Reiter. On inheritance hierarchies with exceptions. In *Proc. AAAI-83*, pages 104–108, 1983.
- [Etherington 87] D.W. Etherington. Formalising non-monotonic reasoning systems. *Artificial Intelligence*, 31:41–85, 1987.
- [Etherington 88] D.W. Etherington. *Reasoning with Incomplete Information*. Research Notes in Artificial Intelligence. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1988.
- [Geffner 88] H. Geffner. On the logic of defaults. In *Proc. AAAI-88*, St. Paul, Minnesota, 1988.
- [Hughes and Cresswell 68] G.E. Hughes and M.J. Cresswell. *An Introduction to Modal Logic*. Methuen and Co. Ltd., 1968.
- [Krauss et al 90] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2), 1990.
- [Konolige 87] K. Konolige. On the relation between default theories and autoepistemic logic. In *IJCAI-87*, pages 394–401, 1987.
- [Kautz and Selman 89] H.A. Kautz and B. Selman. Hard problems for simple default logics. In *Proc. KR-89*, pages 189–197, Toronto, Ont., 1989.
- [McCarthy 80] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.
- [McDermott 82] D.V. McDermott. Nonmonotonic logic ii: Nonmonotonic modal theories. *JACM*, 29(1):33–57, 1982.
- [McDermott and Doyle 80] D.V. McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.
- [Mercer 88] R.E. Mercer. Using default logic to derive natural language suppositions. In *CSCSI-88*, pages 14–21, 1988.
- [Moore 85] R.C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.
- [Neufeld and Poole 88] E. Neufeld and D. Poole. Probabilistic semantics and defaults. In *Fourth Workshop on Uncertainty in Artificial Intelligence*, pages 275–282, 1988.
- [Perrault 87] C.R. Perrault. An application of default logic to speech act theory. Technical Report CSLI-87-90, Stanford University, 1987.
- [Poole 88] D.L. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36(1):27–48, 1988.
- [Reiter and Crisculo 81] R. Reiter and G. Crisculo. On interacting defaults. In *Proc. IJCAI-81*, pages 270–276, Vancouver, B.C., 1981.
- [Reiter 80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
- [Reiter 87] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–96, 1987.
- [Shoham 87] Y. Shoham. A semantical approach to nonmonotonic logics (extended abstract). In *Symposium on Logic in Computer Science*, pages 275–279, Ithaca, New York, 1987.

Plan-Based Terminological Reasoning

Premkumar T. Devanbu Diane J. Litman
 Artificial Intelligence Principles Research Department
 AT&T Bell Laboratories
 600 Mountain Avenue
 Murray Hill, New Jersey 07974*

Abstract

CLASP (Classification of Scenarios and Plans) is a knowledge representation system that extends the notions of subsumption and classification from frame-based languages to plans. The CLASP representation language provides description-forming operators that specify temporal and conditional relationships between actions represented in CLASSIC (a current classification-based knowledge representation language). Subsumption in CLASP builds on term subsumption in CLASSIC and illustrates how term subsumption can be exploited to serve special needs. In particular, the CLASP algorithms for plan subsumption integrate work in automata theory with work in term subsumption. We are using CLASP to store and retrieve information about feature specifications and test scripts in the context of a large software development project.

1 INTRODUCTION

Terminological knowledge representation systems have provided representational support in many areas of Artificial Intelligence (e.g., diagnosis, natural language processing, program synthesis, user interfaces, expert systems). Central to terminological approaches are the interpretation of frames as descriptions, the use of classification and term subsumption inferences to organize frame taxonomies, and differentiation between terminological and assertional aspects of knowledge. A major limitation of current terminological systems, however, is an inability to represent and reason with *plans*. Plans, temporal compositions of actions that

achieve given goals, play a central role in many areas that use terminological knowledge representation systems (natural language processing [Sidner, 1985], expert systems [Swartout and Neches, 1986], user interfaces [Feiner and McKeown, 1990]). While the *generation* and *recognition* of plans has been the focus of much research in automatic reasoning, the knowledge representation task of *managing* collections of plans has largely been unaddressed.

Terminological representation systems have also been used to build information systems to assist in software development. As the size, cost, and lifetime of software systems continue to grow, it becomes increasingly important to find ways to help programmers understand and maintain these systems [Brooks, 1987, Corbi, 1989]. Fischer and Schneider [Fischer and Schneider, 1984] suggest that a large software project should use a knowledge base to collect and disseminate information about all aspects of the system under construction, including specification, architecture, and design. The LASSIE software information system [Devanbu *et al.*, 1990] pursued this idea, using the terminological knowledge representation language CLASSIC [Brachman *et al.*, 1990] to describe the architecture, domain model, and code of the AT&T DefinityTM 75/85¹. CLASSIC is a KL-ONE-like [Brachman and Schmolze, 1985] knowledge representation language, computing terminological inferences such as *classification* and *term subsumption*².

By using a terminological language such as CLASSIC, LASSIE can organize descriptions into a taxonomy and do retrieval based on the semantics of the query and the stored descriptions. A taxonomy of the *actions* and *objects* in the telephony domain forms the core of the LASSIE knowledge base. Unfortunately, LASSIE has representational needs beyond the capabilities of CLASSIC. Because terminological languages (includ-

*Authors listed in alphabetic order only. Prem Devanbu is also with the Dept. of Computer Science, Rutgers University, New Brunswick, NJ. Diane Litman is currently with the Dept. of Computer Science, Columbia University, New York, NY 10027.

¹Definity 75/85 is a software-controlled scalable Private Branch Exchange (PBX) switching product.

²We assume the standard interpretation for these terms, as described in [Brachman *et al.*, 1990].

ing CLASSIC) have no meaningful³ way to represent or reason with *plans*, LASSIE can not be used to describe sequences of actions that achieve particular goals (e.g., Call Forwarding, as found in most modern telephone systems). Context-dependent, causal, temporal, and other relationships cannot be captured. As we will see, plan-like knowledge is important to the entire range of activities associated with software development.

Motivated by such issues, we have designed and implemented a plan-based knowledge representation system called CLASP (CLASification of Scenarios and Plans). CLASP is designed to represent and reason with large collections of plan descriptions, much in the same way current terminological systems reason with object descriptions. CLASP creates plan descriptions from action and object descriptions, using a restricted plan language containing temporal and conditional operators. CLASP uses the semantics of these descriptions to associate plan concepts with sets of plan individuals, and to organize plan concepts into taxonomies based solely on subsumption inferences. CLASP is designed as a companion to the CLASSIC [Brachman *et al.*, 1990] terminological knowledge representation system, and makes intimate use of CLASSIC's well-defined subsumption inference. In particular, the CLASP algorithms for plan subsumption use results in automata theory to extend term subsumption. Indeed, one contribution of our work is to demonstrate how term subsumption can serve as a springboard for implementing special purpose representation and reasoning mechanisms.

In the next section we motivate CLASP by illustrating the importance of plan-like knowledge in programming activity. We then describe the details of the CLASP system, discussing both the representation language and the inference mechanisms. Finally, we show several examples of the use of CLASP in the programming domain, relate CLASP to existing research in taxonomic plan reasoning, and conclude with some thoughts on future directions.

2 WHY PLAN KNOWLEDGE?

The relationship between plans and software information has been explored most widely in the context of automatic and computer-assisted programming [Rich, 1981]. Plans allow the representation of the sequence

³Temporal information such as sequence can be encoded using special role naming conventions (the first step of a plan is stored in a role called "step1," the second in a role called "step2," and so on), with an application program using the conventions to assign the roles the appropriate meaning in a post-hoc manner. This is in fact how plans are represented in the terminological language LOOM [MacGregor, 1988] in the COMET [Feiner and McKeown, 1990] project. What is needed is a principled way to explicitly represent temporal and other planning relations within the representation system.

of events in a program and the assertion of pre and post conditions that are applicable at various points. Besides program synthesis, plans are also useful for program understanding [Johnson and Soloway, 1981, Letovsky, 1987]. Research has shown that plan-like knowledge structures can serve as effective models for understanding human programming behavior [Soloway and Ehrlich, 1984, Soloway *et al.*, 1988].

In the domain of telephone switching that was addressed in LASSIE, plan-like structures are particularly useful in representing features such as "call forwarding" and "call waiting." While a full description of a feature describes behavior under differing conditions, a feature is often illustrated in terms of a *scenario* representing just one aspect of the behavior. Thus, a scenario illustrating one successful use of "call waiting" might be: "*A picks up the phone, gets dial-tone and dials B; since B is off-hook, A gets a special ringing tone and B gets a call-waiting signal; B flashes hook and connects with A.*" A full feature description is a generic description of actions and associated goals, much like a plan, while a scenario is a specific manifestation much like a plan execution trace.

Plan-like structures are also used during the testing phase of software development. In switching software, tests are usually represented as *test scripts*. Test scripts specify stimuli to the switch along with expected responses, for example, "*Pick up the phone; the system should produce a dial-tone.*" A test script is thus representationally very much like a scenario description described above, i.e., a plan-like series of actions. Given the number of test scripts — a large project can have on the order of 10,000 scripts — the ability to represent and manage such scripts within a LASSIE-like system would be extremely useful.

Plans can also be useful in explaining the behavior of distributed software systems, like the AT&T DefinityTM 75/85 switch. The processing of a typical stimulus to a switch involves the exchange of messages between several processes. These messages are logged to a file and examined off-line. The comprehension of these message traces is an important step in understanding the software, and involves constructing explanations that are very plan-like. For example, a message trace might show a request to a trunk handling process to open a connection to another switch; this might be followed by a series of messages requesting packet transmission, followed by another message to close the transmission. The explanation of this trace would be a plan with a sequence of actions: the first action connects the trunk and makes it ready for transmission; a series of actions send the messages; a final action closes the connection. It would be useful to create and store such explanations for subsequent use.

3 CLASP

CLASP is a plan-based knowledge representation system that is a “complement” to CLASSIC, the term subsumption system used in LASSIE. That is, just as CLASSIC allows users to define descriptions and create instances of terms, CLASP allows users to define *plan* concepts and create *scenario* instances. As we will see, CLASP plans are compositionally defined from CLASSIC terms. Similarly, just as subsumption and classification are the central inferences in CLASSIC, plan subsumption and classification are the core inferences in CLASP. In particular, CLASP can compute the generalization relationships that organize plan concepts into taxonomies, and can associate plan concepts with sets of scenarios (plan individuals). The plan-based inferences of CLASP use results in automata theory to extend the capabilities already available in CLASSIC⁴.

3.1 REPRESENTATION OF ACTIONS

CLASP complements the term language of CLASSIC by providing plan operators to form plans from CLASSIC terms. In particular, actions are the building blocks of CLASP plans and scenarios. Since actions can be represented adequately in current terminological systems such as CLASSIC, the representation of actions in CLASSIC is briefly discussed here.

Frames in CLASSIC are called *concepts*. They are (potentially complex) descriptions and are formed by restricting other descriptions using a small set of description-forming operators. For example, existing concepts can be conjoined using the operator “AND.” Roles can also be further constrained. The “ALL” value restriction restricts all fillers of a particular role to be of a certain type, while the number restrictions “AT-LEAST” and “AT-MOST” specify constraints on the number of fillers for a particular role. “FILLS” specifies particular individuals that fill the role. While there are other types of restrictions in CLASSIC, they will not be needed to understand the examples in this paper. *Individuals* are specific instances of concepts, and are created using the same restrictions as for concepts. The extension of a concept is the set of individuals described by the concept.

CLASP uses CLASSIC to define built-in concepts *Action*, *State*, and *Agent*. For example, CLASP defines the CLASSIC concept *Action* to represent a STRIPS action operator [Fikes and Nilsson, 1971], where role restrictions specify the characteristics of the roles *ACTOR*, *PRECONDITION*, *ADD-LIST*, *DELETE-LIST*, and *GOAL*⁵.

⁴The use of terminological models to enhance other paradigms has also been applied to production systems [Yen *et al.*, 1989].

⁵In our informal notation, *Concept-NAMES* will be shown in capitalized typewriter font, *individual-names* in lower case typewriter font, *ROLE-NAMES* in upper case typewriter

In particular, an action is defined thus:

```
(DEFINE-CONCEPT
  Action
  (PRIMITIVE
    (AND Classic-Thing
      (AT-LEAST 1 ACTOR)
      (ALL ACTOR Agent)
      (ALL PRECONDITION State)
      (ALL ADD-LIST State)
      (ALL DELETE-LIST State)
      (ALL GOAL State))))
```

The above definition states that “An Action is a *Classic-Thing*, with at least one *ACTOR*, all of whose *ACTORS* are of type *Agent*, all of whose *PRECONDITIONS* are of type *State*, all of whose *ADD-LISTS* are of type *State*, all of whose *DELETE-LISTS* are of type *State*, and all of whose *GOALS* are of type *State*” (assuming the previous definition of the concepts *Classic-Thing*, *State* and *Agent* in CLASSIC). This example has the role restrictions that there must be at least one filler of the role *ACTOR*, that all fillers of the role *ACTOR* must be of type *Agent*, and so on. The *PRIMITIVE* operator is used to specify that the concept definition is not necessary and sufficient, that is, that the concept cannot be automatically placed into a concept taxonomy via subsumption and classification inferences⁶.

Actions can be restricted to define various specialized actions in the LASSIE domain. As we will see, these action concepts can be combined in CLASP to form plans, while the individual instances of the actions can be combined to form scenarios. For example, the *Action* concept specializes into *System-Acts* and *User-Acts*. *System-Act* is defined below:

```
(DEFINE-CONCEPT
  System-Act
  (AND Action
    (ALL ACTOR System-Agent)))
```

This definition declares that *System-Act* is a subconcept of *Action*, where all the fillers of the (inherited) role *ACTOR* are restricted to be individuals described by the concept *System-Agent* (which itself must be defined in CLASSIC as a subconcept of *Agent*). Unlike the primitive concept *Action*, this concept is fully specified by necessary and sufficient conditions.

font, and CLASSIC-OPERATORS (and later CLASP-OPERATORS) in upper case.

⁶Doyle and Patil [Doyle and Patil, 1989] argue against restricting knowledge representation languages in order to support efficient subsumption (as is done in CLASSIC). In particular, they argue that subsumption should not be given such a central role. For example, in their own work they found that a large percentage of concept definitions could not be specified in terms of necessary and sufficient conditions.

The definition of **System-Act** can itself be restricted⁷:

```
(DEFINE-CONCEPT
  Connect-Dialtone-Act
  (AND System-Act
    (EXACTLY 1 PRECONDITION)
    (ALL PRECONDITION
      (AND Off-Hook-State
          Idle-State)))
    (EXACTLY 1 ADD-LIST)
    (ALL ADD-LIST Dialtone-State)
    (EXACTLY 1 DELETE-LIST)
    (ALL DELETE-LIST Idle-State)
    (EXACTLY 1 GOAL)
    (ALL GOAL
      (AND Off-Hook-State
          Dialtone-State))))
```

Informally, the system performs a **Connect-Dialtone-Act** and generates a dialtone after a user picks up a phone. Notice that this concept is defined by specifying more properties that restrict **System-Act**.

CLASSIC can also be used to create individuals that are described by concepts. In particular, an individual must satisfy the restrictions of the describing concept. The following **CLASSIC** function creates an individual **act1** that is described by the concept **System-Act** (defined above):

```
(CREATE-IND
  act1
  (AND System-Act
    (FILLS ACTOR switching-system)))
```

Note that for **act1** to satisfy the restrictions of **System-Act**, the specified filler of the role **ACTOR** (the individual **switching-system**) must itself have been previously created, and must be describable by the concept **System-Agent**.

3.2 REPRESENTATION OF PLANS

CLASP provides a representation language for *plans*, descriptions that organize and group together context-independent **CLASSIC** action descriptions. Thus, a **CLASSIC** action type such as **Connect-Dialtone-Act** will occur in many **CLASP** plans. As we will see, by being defined in terms of **CLASSIC** actions, the algorithms for plan classification can take advantage of **CLASSIC** inheritance and subsumption.

A user defines a plan in **CLASP** by specifying a name and restricting the roles **PLAN-EXPRESSION** (a plan

⁷Recall that **AND** is a **CLASSIC** operator that conjoins concepts. **EXACTLY** (followed by a number) is our notation for an operator that defines precisely how many fillers are allowed for a slot. In **CLASSIC** this would be expressed using **AT-MOST** and **AT-LEAST** operators with the same number.

concept expression, specified using the syntax below), and optional roles **INITIAL** and **GOAL**. Plan concept expressions are compositionally defined from action and state concepts using the plan description forming operators **SEQUENCE**, **LOOP**, **REPEAT**, **TEST**, **OR**, and **SUBPLAN**:

```
<plan-concept-expression> ::= 
  <action-concept> | 
  (SEQUENCE <plan-concept-expression>+) | 
  (LOOP <plan-concept-expression>) | 
  (REPEAT <integer>
    <plan-concept-expression>) | 
  (TEST (<state-concept>
    <plan-concept-expression>)+) | 
  (OR <plan-concept-expression>+) | 
  (SUBPLAN <symbol>)
```

In other words, plan definitions restrict the type, the (conditional) presence, and sequential⁸ temporal ordering of action individuals in scenarios (plan individuals) described by the plan. (**<action-concept>** and **<state-concept>** refer to **CLASSIC** concepts subsumed by the concepts **Action** and **State**. Recall that **Action** and **State** are pre-defined in **CLASSIC** by **CLASP**). The following examples illustrate the interpretation of the constructs listed above:

- (**SEQUENCE A B C**): An action of type **A** is followed by an action of type **B**, which is followed by an action of type **C**.
- (**LOOP A**): Zero or more actions of type **A**.
- (**REPEAT 7 A**): Equivalent to (**SEQUENCE A A A A A A A**).
- (**TEST (S1 A) (S2 B)**): If the current state is of type **S1**, then action type **A**, else if state type **S2** then action type **B**.
- (**OR A B**): Either action type **A** or type **B**⁹.
- (**SUBPLAN Plan-Name**): Syntactically insert **Plan-Name**'s plan expression¹⁰.

The root of the plan taxonomy is the built-in **CLASP** concept **Plan**, where

```
(DEFINE-PLAN
  Plan
  (PRIMITIVE
    (AND Clasp-Thing
      (ALL INITIAL State)
      (ALL GOAL State)))
```

⁸We can also specify partial orders, using the operator **OR** to explicitly specify that any number of sequential descriptions are acceptable.

⁹Our algorithm for subsumption currently assumes that plans are expressed nondeterministically. Note that this means that the sets described by concepts **A** and **B** must be disjoint.

¹⁰We prohibit recursive plan definitions.

```
(EXACTLY 1 PLAN-EXPRESSION)
(ALL PLAN-EXPRESSION
  (LOOP Action))))
```

We can use the plan taxonomy to represent and organize descriptions of Definity 75/85 features (recall Section 2). For example, the following are informal CLASP definitions representing an abstract view of POTS (the default feature or "Plain Old Telephone Service"), as well as another plan used in the POTS definition:

```
(DEFINE-PLAN
  Pots-Plan
  (AND
    Plan
    (ALL PLAN-EXPRESSION
      (SEQUENCE
        (SUBPLAN
          Originate-And-Dial-Plan)
        (TEST
          (Callee-On-Hook-State
            (SUBPLAN Terminate-Plan))
          (Callee-Off-Hook-State
            (SEQUENCE
              Non-Terminate-Act
              Caller-On-Hook-Act
              Disconnect-Act)))))))
  (DEFINE-PLAN
    Originate-And-Dial-Plan
    (AND
      Plan
      (ALL PLAN-EXPRESSION
        (SEQUENCE
          Caller-Off-Hook-Act
          Connect-Dialtone-Act
          Dial-Digits-Act)))))
```

Of course, there must also be definitions for every other action, state and plan in these definitions (defined using CLASSIC and CLASP, respectively). For example, the definition of *Connect-Dialtone-Act* in CLASSIC was presented earlier. Informally, *Pots-Plan* describes a plan in which the caller picks up a phone, gets a dialtone, and dials a callee. If the callee's phone is on-hook, the call goes through; if the callee's phone is off-hook, the caller gets a busy signal, hangs up, and is disconnected.

Note that with the exception of TEST and SUBPLAN, plan expressions built using the CLASP operators correspond to regular expressions. CLASP, in fact, can transform such plan representations into regular expressions. First, CLASP can use the semantics underlying its action representation to replace all TEST operators with expressions involving OR operators and new action descriptions. This is because the preconditions of an action must describe the state of the world before an individual described by the action can be successfully executed. Similarly, in each (*State*

Action) argument of a TEST operator, the state describes the state of the world that must be satisfied before an action individual can be described by the action. Thus, for example, (TEST (*State1 Action1*) (*State2 Action2*)) is equivalent to (OR *Action1b Action2b*), where *Action1b* specializes *Action1* by specifying that the type restriction of PRECONDITION of *Action1b* is the conjunction of type *State1* and the PRECONDITION type restriction of *Action1*. The construction of *Action2b* is similar¹¹. CLASP will construct and define such action descriptions, in order replace all expressions involving the operator TEST with equivalent expressions involving the operator OR. As for SUBPLAN, recall that this construct was just a notational convenience. The ability to perform this transformation will have important implications with respect to plan subsumption. In particular, plan subsumption will be able to use results in automata theory to extend term subsumption.

Although plans are defined within CLASP, they are internally represented using CLASSIC. This can be done because CLASSIC provides hooks for storing information (e.g., via Common LISP types such as lists) that CLASSIC itself cannot meaningfully represent. CLASP constructs a finite automaton recognizing the plan expression, uses an array to represent this automaton, and stores the array using a CLASSIC role restricted to fillers of LISP type *Array*. Note that although "represented," the planning information is outside the scope of the CLASSIC classification and subsumption processes. Instead, CLASP constructs a plan taxonomy using its own plan subsumption algorithms, as described below.

3.3 REPRESENTATION OF SCENARIOS

A CLASP *plan individual* is called a *scenario*. As in classical planners such as STRIPS [Fikes and Nilsson, 1971], a plan individual corresponds to a sequence of actions that when executed in an initial state achieves a goal state¹². A scenario is created in CLASP by asserting that it is described by (a specialization of) the concept *Plan*, by specifying the individuals that satisfy the restrictions on the plan's initial state as well as goal state, and by specifying a scenario-expression that is described by the plan's plan-expression. (The initial state is the state in which the scenario begins to execute, the goal state is the state that scenario

¹¹This construction will also cover the case when a complex plan expression takes the place of simple action descriptions such as *Action1b*. This is because the complex expressions are "unbundled" into such simpler forms when constructing finite state machines, as discussed below.

¹²Like STRIPS, the temporal ordering of action sequences in scenarios must be total; in other words, CLASP scenarios are *linear*. Note that while planners such as STRIPS *synthesize* plans, the CLASP knowledge representation *manages* plans.

execution achieves, and the actions are the ordered sequence of action individuals constituting the scenario.) In particular, scenario expressions are built from CLASSIC action individuals:

$\langle \text{scenario-expression} \rangle ::= (\langle \text{action-individual} \rangle +)$.

The following is an example of a scenario:

```
(CREATE-SCENARIO
  pots-busy-scenario
  (AND Plan
    (FILLS INITIAL state-u1on-u2off)
    (FILLS GOAL state-u1on)
    (FILLS PLAN-EXPRESSION
      (caller-off-hook-u1
       connect-dialtone-on-u1
       dial-digits-u1-to-u2
       non-terminate-on-u2
       caller-on-hook-u1
       disconnect-u1))).
```

With the callee off-hook in the initial state, this scenario represents the case where a user picks up the phone, the system generates a dialtone, the user dials the callee, the system generates a busy signal (the call failed to terminate), the user hangs up, and the system disconnects. More precisely, we define *pots-busy-scenario* to be a *Plan* whose *INITIAL* is filled by *state-u1on-u2off*, whose *GOAL* is filled by *state-u1on*, and whose *PLAN-EXPRESSION* is filled by a sequence of action individuals that begins with *caller-off-hook-u1* and continues through to *disconnect-u1*. The individuals used here, such as *state-u1on-u2off* (the initial state in which the caller is on-hook and the callee is off hook) and actions such as *caller-off-hook-u1* (the caller goes off-hook, i.e. picks up the phone) are assumed to have been previously defined in CLASSIC¹³.

When a scenario is created, CLASP confirms that the given sequence of actions will indeed transform the specified initial state into the goal state (i.e., that the scenario is well-formed). During this process any unspecified intermediate states (the fillers of the precondition and goal roles of each action individual) are inferred, using the STRIPS rule. While such information is often computed during plan synthesis, typically it is not stored in the final plan¹⁴. As will be seen below, CLASP needs such information to determine if a scenario is described by a plan. Informally, the STRIPS rule specifies that when an action is applied in a particular world state, the new

¹³For clarity, we reflect the type of an individual in its name; e.g., *state-u1on-u2off* is an individual of type *State*; also, *u1* refers to the caller and *u2* to the callee, etc.

¹⁴Work on plan reuse [Kambhampati and Hendler, 1989] has shown a similar need for the maintenance of such information.

world state satisfies every state description in the *ADD-LIST* and every previously satisfied state description not in the *DELETE-LIST*. For example, assume *state-u1on-u2off* in *pots-busy-scenario* is described by

$(\text{AND State-U1on State-U2off}),$

and that fillers of *ADD-LIST* and *DELETE-LIST* of the action concept *Caller-Off-Hook* are restricted to $(\text{AND Idle-State State-U1off})$ and *State-U1on*, respectively. The STRIPS rule computes that the state of the world after *caller-off-hook-u1* is described by

$(\text{AND Idle-State}$
 $\quad \text{State-U1off}$
 $\quad \text{State-U2off}).$

Scenarios are described by (are instances of) plans. A scenario described by a plan is a member of the class (the set of scenarios) corresponding to the plan. Intuitively, a scenario is an instance of a plan if the temporal and conditional restrictions used to define the plan concept are satisfied in the scenario. As with plans, CLASSIC can represent but not classify scenarios. In particular while CLASSIC can determine if the scenario fillers of *INITIAL* and *GOAL* meet the plan's restrictions, it cannot determine this with respect to *PLAN-EXPRESSION*. As will be seen, CLASP determines if a sequence of action individuals is described by a plan expression (a "grammar" of action descriptions) by parsing, in conjunction with term subsumption. For example, CLASP will determine that *pots-busy-scenario* is described by *Pots-Plan*.

3.4 SUBSUMPTION OF SCENARIOS

In this section we present the algorithm for *scenario subsumption*¹⁵, which computes whether a plan describes a scenario. Scenario *classification* uses subsumption to determine all plans that a scenario satisfies. Scenario subsumption enables CLASP to explicitly assert CLASSIC instance relationships between plans and scenarios. Intuitively, a plan *P* is satisfied by a scenario *s* if *P* describes *s*, that is, if the restrictions defining *P* are satisfied by *s*. Recall that every plan has restrictions concerning the roles *INITIAL*, *GOAL*, and *PLAN-EXPRESSION*. (Henceforth we will use *o-subsumes*, *s-subsumes*, and *p-subsumes* to refer to object subsumption as supported in current terminological systems such as CLASSIC, CLASP subsumption between *PLAN-EXPRESSION* restrictions and fillers, and CLASP subsumption between *PLAN-EXPRESSION*

¹⁵As in CLASSIC, we use the term "subsumption" rather than the term "realization" to describe this inference. We will also say, informally, that a plan *P* *describes* a scenario *s* (or that *s* *satisfies* *P*) when *P* subsumes *s*.

restrictions, respectively¹⁶.) Since the restrictions regarding INITIAL and GOAL are CLASSIC restrictions, s satisfies P if INITIAL and GOAL of s are o-subsumed by INITIAL and GOAL of P , and if PLAN-EXPRESSION of s is s -subsumed by PLAN-EXPRESSION of P .

Informally, a plan expression of a scenario (call it $s\text{-exp}$) is s -subsumed by that of a plan (call it $P\text{-exp}$) if the action individuals of $s\text{-exp}$ are o-subsumed by the action descriptions that constitute $P\text{-exp}$ subject to the temporal and conditional restrictions specified by the CLASP operators of $P\text{-exp}$. More formally, $s\text{-exp}$ is s -subsumed by $P\text{-exp}$ if $s\text{-exp}$ is a string in the language defined by $P\text{-exp}$. Let Σ be the alphabet consisting of all concepts of type Action, and Σ_i^* denote all strings of action individuals, i.e., strings of individuals where each individual is described by a symbol of Σ . A plan expression over Σ denotes the set of strings from Σ_i^* in the language generated by the plan expression. $P\text{-exp}$ s -subsumes $s\text{-exp}$ if $s\text{-exp}$ is in the subset of Σ_i^* denoted by $P\text{-exp}$. For example, with $P\text{-exp}$ equal to

```
(SEQUENCE (OR Caller-On-Hook-Act
          Callee-On-Hook-Act)
          Hangup-Act)
```

and $s\text{-exp}$ equal to

```
(caller-on-hook1 busy-hangup10),
```

$P\text{-exp}$ s -subsumes $s\text{-exp}$ if Caller-On-Hook-Act o-subsumes caller-on-hook1 and Hangup-Act o-subsumes busy-hangup10. Note that the use of o-subsumption enables the recognition of action individuals that are directly as well as abstractly described by the action terms in $P\text{-exp}$. In contrast, the above plan expression would not subsume the scenario expression (busy-hangup10).

Since CLASP plan expressions can be transformed into regular expressions, s -subsumption can be efficiently implemented. To test whether $s\text{-exp}$ is s -subsumed by $P\text{-exp}$, CLASP tests whether $s\text{-exp}$ is accepted by the finite automaton recognizing the language denoted by $P\text{-exp}$. (CLASP builds a finite automaton whenever a plan is defined). Because each transition in the finite automaton corresponds to an o-subsumption rather than equality check, we call our automata *Extended Finite Automata* (EFA). The particular pattern matching algorithm used in CLASP is $O(mn)$ o-subsumptions in the worst case, where m is the size of the finite-state

¹⁶An orthogonal plan taxonomy could also be organized via goals using *g-subsumption*. Informally, one plan g-subsumes another if the fillers of the GOAL and INITIAL roles satisfy o-subsumption relationships; g-subsumption thus ignores any relationships (i.e., s -subsumption and p-subsumption) among plan expressions, and simply checks the conditions in the world before and after the execution of the plan.

machine equivalent of the plan and scenario is of size n [Sedgewick, 1988]. The analysis of o-subsumption will depend on the particular terminological model used. In CLASSIC, determining whether an individual satisfies a description (no embedded defined concepts) is unknown, while determining whether an individual matches a concept is believed to be NP-hard or NP-complete [Patel-Schneider and Borgida, 1991]. However, if the plans and scenarios are constructed from a stable CLASSIC knowledge base of action concepts and instances, CLASSIC can be used to pre-compute and cache all the o-subsumptions between concepts and individuals in a taxonomy, for later use by CLASP. Using o-subsumption results that are computed and cached “off-line” would allow the complexity of s -subsumption to be simply $O(mn)$.

Again, we emphasize here that the CLASP algorithm performs o-subsumption rather than equality checking between action concepts in Σ and action individuals in Σ_i . This integration of term subsumption with regular expression processing provides a powerful facility for retrieving scenarios using incomplete and abstract plan descriptions. For example, the root of the plan description taxonomy, (LOOP Action), would s -subsume (and thus retrieve) every scenario.

3.5 SUBSUMPTION OF PLANS

Plan classification is the determination of all plans that are more general and all plans that are more specific than a given plan. Plan classification organizes plans (classes) into a taxonomy according to the subset relation. Plan classification is based on *plan subsumption*, determination of whether one plan is more general than another. A plan, P is more general than a plan Q if any scenario that satisfies Q necessarily also satisfies P . o-subsumption can be used to determine generality for descriptions filling the plan roles INITIAL and GOAL. CLASP, however, must provide a notion of *p-subsumption* to determine PLAN-EXPRESSION generality.

For plan expressions PE1 and PE2, PE1 p-subsumes PE2 if the language described by PE1 is necessarily a superset of that described by PE2. If L1 is the set of scenarios satisfying PE1, and L2 the set satisfying PE2, PE2 is p-subsumed by PE1 if $L2 \subseteq L1$. For example, given action descriptions A, B, and C, where A only o-subsumes C,

```
(SEQUENCE (LOOP A) (OR B A))
```

p-subsumes (SEQUENCE A C B) but does not p-subsume (SEQUENCE B A). The root of the plan taxonomy, (LOOP Action), p-subsumes every plan.

P-subsumption can be understood in terms of an extension of regular expression subsumption. Given two plan expressions PE1 and PE2, we compute the equivalent deterministic finite automata, EFA¹ and

EFA^2 , and their Cartesian product EFA^χ . The states of EFA^χ are ordered pairs of the form $\langle s_i^1, s_i^2 \rangle$, where s_i^j is one of the states of the machine EFA^j , $j = 1, 2$; EFA^χ would reach state $\langle s_i^1, s_i^2 \rangle$ after scanning through a scenario S just in case the machine EFA^j would be in state s_i^j , $j = 1, 2$ after scanning through the same scenario S .

The Cartesian product machine helps us determine if all the scenarios accepted by one EFA are also accepted by the other (i.e., if one EFA subsumes the other). This is accomplished by looking in the product machine for states where one of the machines accepts and the other rejects, as well as states where both accept. If a state of the form $\langle \text{accept}, \text{reject} \rangle$ occurs, that means there is a scenario where EFA^1 accepts and EFA^2 rejects. Now, if there are states of the form $\langle \text{accept}, \text{accept} \rangle$, $\langle \text{reject}, \text{accept} \rangle$, and no states of the form $\langle \text{accept}, \text{reject} \rangle$, then clearly, all scenarios accepted by EFA^1 are also accepted by EFA^2 (and EFA^2 accepts additional scenarios); thus, in this case, PE1 is subsumed by PE2. Likewise, if the product machine contains states of the form $\langle \text{accept}, \text{accept} \rangle$, $\langle \text{accept}, \text{reject} \rangle$, and no $\langle \text{reject}, \text{accept} \rangle$ states, then the first plan subsumes the second. If there are only states of the form $\langle \text{accept}, \text{accept} \rangle$, the two plans accept the same language. We can also distinguish between cases where there are not subsumption relationships between the two plans. If there are $\langle \text{accept}, \text{reject} \rangle$, $\langle \text{reject}, \text{accept} \rangle$, and $\langle \text{accept}, \text{accept} \rangle$ states, PE1 intersects PE2. That is, while neither plan subsumes the other, it is possible to have a scenario that could be described by both plans. Otherwise, the two plans are disjoint, as there can never be a scenario that is described by both plans. For example, consider the following plan expressions, where concepts A and B are disjoint, and concept A1 specializes A:

P1: (LOOP A)
P2: (OR A1 B)
P3: (SEQUENCE A1 A1)

The results of p-subsumption will determine that P1 intersects P2, P1 subsumes P3, and P2 is disjoint from P3.

The construction of EFA^χ , however, is not simple. Recall that the transitions in $EFA^{1,2}$ involve o-subsumption tests, not equality. Figure 1 illustrates the complexity of computing the product when transitions involve o-subsumption tests. Portions (a) and (b) of the figure each show a sample machine consisting of one transition. Portion (c) shows the corresponding product machine assuming the transitions are based on the standard equality tests (note that the states are now pairs of individual states in the original machines). Portion (d), in contrast, shows the cross product machine if the transitions are instead interpreted as o-subsumption tests. In particular, here A and B

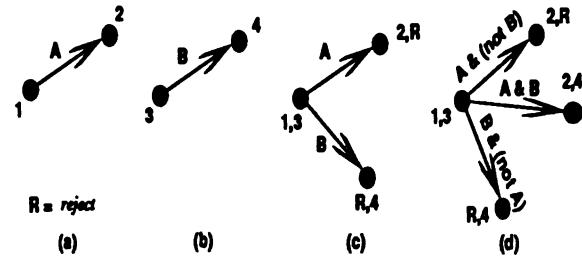


Figure 1: Constructing the Cartesian Product Machine

denote intersecting CLASSIC action descriptions¹⁷.

As discussed, p-subsumption, uses CLASSIC o-subsumption from concepts to concepts. o-subsumption for descriptions (no embedded defined concepts) is polynomial, while subsumption allowing defined concepts is believed to be NP-hard or NP-complete.[Patel-Schneider and Borgida, 1991] As with o-subsumption between concepts and individuals, if such results are cached, o-subsumption is a constant time operation once computed. The subsumption of regular expressions is P-SPACE hard [Wellman, 1988]. The intractability of this problem arises from the fact that regular expressions and their equivalent non-deterministic finite state machines are very compact representations. In fact, if they are converted to their equivalent deterministic finite state machines (leading to an exponential increase in the size of the machines) the subsumption can be done in polynomial time. While computing p-subsumption we convert the CLASP plan expressions directly into finite state machines, prior to computing the cross-product. In the LASSIE domain, we find that the resulting finite state machines are generally quite small, and in practice, p-subsumption rarely takes more than a few hundred milliseconds. (It should also be noted, as before, that CLASSIC subsumptions were previously computed and cached; they were essentially constant time operations.)

¹⁷CLASSIC provides tests for subsumption, disjointness and equality between concepts; with these, the meaningfulness of action concepts such as A & B, A & (not B) can be checked for the actions A and B. Though CLASSIC does not support negation, we construct these concepts merely to verify that the Cartesian product machines can reach certain states.

4 USING CLASP

This section illustrates the use of CLASP in enhancing subsumption-based retrieval systems such as LASSIE¹⁸. The examples illustrate the power of combining term subsumption with plan processing. First, the mechanisms for representing and organizing feature descriptions such as Pots-Plan provide a very flexible method for retrieving feature scenarios such as *pots-busy-scenario*. In particular, definitions of plan descriptions containing action “wildcards” can be used to retrieve scenarios satisfying particular planning relationships. The set of scenarios retrieved are just those scenarios that are subsumed by the plan description during CLASP scenario subsumption. For example, a plan expression containing the following plan expression can be used to find all the *contexts* in which specializations of the CLASSIC description *Connect-Dialtone-Act* occur:

```
(SEQUENCE (LOOP Action)
          Connect-Dialtone-Act
          (LOOP Action)).
```

Here, CLASP represents context while CLASSIC adds context-independent action abstraction. Plan descriptions can also be used to retrieve scenarios satisfying *temporal* context relationships:

```
(SEQUENCE (LOOP Action)
          Caller-On-Hook-Act
          Callee-On-Hook-Act).
```

Here CLASP is used to specify that a caller goes on-hook immediately before a callee, and that all other actions in the scenario precede these actions.

We can also retrieve scenarios by specifying restrictions on intermediate states:

```
(SEQUENCE
  (LOOP Action)
  (AND Action
    (ALL PRECONDITIONS Busy-State))
  (LOOP Action)).
```

This plan description is satisfied by any scenario in which a phone is busy at some point; the retrieval is based on information about role fillers of action individuals asserted into the knowledge base by the CLASP state computations.

Finally, *plan* subsumption can support retrieval of feature descriptions rather than scenarios. This capability could be used to determine if any existing features

¹⁸CLASP[Litman and Devanbu, 1990] representation and subsumption is fully implemented in Common Lisp and CLASSIC. To date we have used CLASP to represent and query a small set of feature descriptions and feature scenarios added to a LASSIE knowledge base.

handle a target behavioral description. For example, plan classification could determine that a call forwarding plan description describes all plans in which a user dials the number of one phone, but another phone actually rings.

The capabilities supporting feature processing can also be used to flexibly retrieve test scripts. Suppose a tester wanted to run scripts to ensure that a particular process wrote a report to a log file every time any feature (call forwarding, call waiting, selective call reject) was enabled or disabled at a phone. Because many devices can be considered to be “phones” and not all features operate on these devices, finding all relevant scripts is an extremely difficult task. With CLASP, however, the tester may issue a simple query:

```
(SEQUENCE
  (LOOP Action)
  (OR (AND Feature-Enable-Action
            (ALL HAS-OPERAND Phone))
      (AND Feature-Disable-Action
            (ALL HAS-OPERAND Phone)))
  (LOOP Action)).
```

Again, we see how scenario and term subsumption can be combined to support retrieval given incomplete and abstract descriptions. This query also illustrates how the taxonomic reasoning of CLASSIC can be combined with the regular expression facility of CLASP to help reduce the “cognitive load” on testers. For example, the tester does not have to remember that ISDN stations, ordinary handset telephones, incoming trunks, and many other devices are all considered to be “phones”; additionally, the user doesn’t have to remember all the different features and how they are enabled.

CLASP can be used to query collections of message traces to identify those traces that show a certain behavior. For example, if we wanted to make sure that the switch didn’t allow a particular incoming trunk to make an outgoing call¹⁹, we could check that there were no message traces where a incoming trunk made a call into the system, and subsequently was able to dial out. This could be determined by constructing the following query:

```
(SEQUENCE
  (LOOP Action)
  (AND Off-Hook-Action
    (FILLS ACTOR incoming-trunk1))
  (LOOP Action)
  (AND Connect-Action
    (FILLS ACTOR incoming-trunk1)
    (ALL RECIPIENT Toll-Trunk))
  (LOOP Action)).
```

¹⁹If this were allowed, then it would be possible to dial into a PBX, and then dial out and make a free long-distance call.

If any answers are retrieved, then there are some violations.

5 RELATED WORK

Taxonomic reasoning has largely been the concern of research in knowledge representation, particularly in KL-ONE-like systems. While plan reasoning has typically been the concern of fairly orthogonal research areas, namely plan recognition and plan synthesis, there are a few examples where abstraction in planning plays an important role²⁰.

A plan abstraction hierarchy is central to the plan recognition work of Kautz [Kautz, 1987]. However, in his taxonomy, plan nodes are still terms rather than structures, and the suitability of the representation for computing terminological inferences is not of concern. In the field of plan synthesis, Tenenberg [Tenenberg, 1989] uses a plan hierarchy to construct abstract plan solutions that constrain later search, where any abstract solution can always be specialized by choosing a specialization of each abstract plan step. Thus, while plans in Tenenberg's hierarchy are compositions of actions (like in CLASP), plans must always be structurally isomorphic across abstraction levels.

Several research projects have used existing terminological knowledge representation systems to directly represent and organize plans. Swartout and Neches [Swartout and Neches, 1986] organize plans into a taxonomy by the semantics of the goals achieved (rather than the methods that are used to achieve the goals). In our terminology, the plan hierarchy is thus organized via g-subsumption, rather than by s-subsumption and p-subsumption as in CLASP. The COMET multimodal generation project [Feiner and McKeown, 1990] uses an existing terminological language to "represent" sequential plans. It uses intuitive role-naming conventions that are understood by the generation programs. This information is not understood, however, by the representation system during its subsumption processes.

Finally, a plan hierarchy is central to Wellman's synthesis work [Wellman, 1988], which is in fact the most similar approach to CLASP. Plans are built from actions represented in a term subsumption language, and plan classes are organized into a hierarchy based on a notion of subsumption. However, the actual language for constructing plan descriptions is quite different (largely due to differences in domain); for example, Wellman's language is totally atemporal while CLASP allows the representation of sequence. Correspondingly, the algorithms for classifying and subsuming plan descriptions must differ. Furthermore, Wellman only represents and subsumes plan classes, while

²⁰Plan decomposition hierarchies also play a role in several systems[Sacerdoti, 1977, Kautz, 1987].

CLASP is also concerned with plan individuals.

6 CONCLUSION

CLASP is a knowledge representation and reasoning system that extends the notions of term subsumption to *plans*. In this paper we presented the CLASP language for defining plan descriptions and creating plan individuals that satisfy such descriptions. As in subsumption-based systems, descriptions and individuals are organized into taxonomies based on subsumption and classification inferences. We have discussed several subsumption inferences relevant to plans — s-subsumption, p-subsumption, and g-subsumption — and have shown how such inferences are related to a core notion of term subsumption. We have motived our work by demonstrating the importance of managing large collections of plans. In particular, we have used CLASP to provide a viable data model as well as a framework for representing and retrieving information in software information systems.

Several extensions to CLASP would be particularly useful. Although the current representation language can encode an interesting class of plans in the telephony domain, the language is not as general as many representations used in plan synthesis. For example, the standard operators for composing plans not only include sequence and choice, but also iteration, recursion, and concurrency [Georgeff, 1987]. However, to extend the expressive power (for example, allowing the representation of parallel actions), new algorithms for plan subsumption would be needed. One might also add inheritance and an elementary notion of assertions to CLASP. Assertions would allow the definition of constraints on execution patterns and could be used during software debugging to check execution traces for anomalous behavior.

Acknowledgements

We would like to thank Bruce Ballard, Ron Brachman, David Etherington, Alan Frisch, Henry Kautz, Peter Patel-Schneider, Martha Pollack, and Marc Vilain for their comments on drafts of this paper. We would also like to thank Jim Piccarello and Ian Bruce for useful discussions regarding the application of knowledge representation to Definity 75/85.

References

- [Brachman and Schmolze, 1985] R. J. Brachman and J. G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
- [Brachman *et al.*, 1990] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with classic: When and

- how to use a kl-one-like language. In J. Sowa, editor, *Formal Aspects of Semantic Networks*. Morgan Kaufmann, 1990.
- [Brooks, 1987] F. P. Brooks. No silver bullet: Essence and accidents of software engineering. *IEEE Computer Magazine*, April 1987.
- [Corbi, 1989] T. A. Corbi. Program understanding: A challenge for the 1990's. *IBM Systems Journal*, 28(2), 1989.
- [Devanbu *et al.*, 1990] P. Devanbu, R. J. Brachman, P. G. Selfridge, and B. W. Ballard. Lassie: a knowledge-based software information system. In *Proc. 12th International Conference on Software Engineering*, Nice, France, April 1990.
- [Doyle and Patil, 1989] Jon Doyle and Ramesh S. Patil. Language restrictions, taxonomic classification, and the utility of representation services. Technical Report MIT/LCS/Technical Memo 387, MIT, 1989.
- [Feiner and McKeown, 1990] S. Feiner and K. McKeown. Generating coordinated multimedia explanations. In *Proceedings of the 6th Conference on Artificial Intelligence Applications*, pages 290–296, Santa Barbara, CA, March 1990.
- [Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4):189–208, 1971.
- [Fischer and Schneider, 1984] G. Fischer and M. Schneider. Knowledge-based communication processes in software engineering. In *Proceedings, Seventh International Software Engineering Conference*, Orlando, FL, 1984.
- [Georgeff, 1987] M. P. Georgeff. Planning. In J. F. Traub, B. J. Grosz, B. W. Lampson, and N. J. Nilsson, editors, *Annual Review of Computer Science*, pages 359–400. Annual Reviews Inc, 1987.
- [Johnson and Soloway, 1981] W. L. Johnson and Elliot Soloway. Proust: Knowledge-based program understanding. *IEEE Transactions on Software Engineering*, 11(3), March 1981.
- [Kambhampati and Hendler, 1989] S. Kambhampati and J. A. Hendler. Flexible reuse of plans via annotation and verification. In *Proceedings of the 5th IEEE Conference on Applications of AI*, 1989.
- [Kautz, 1987] H. A. Kautz. *A Formal Theory of Plan Recognition*. PhD thesis, University of Rochester, Rochester, NY, May 1987.
- [Letovsky, 1987] S. Letovsky. Program understanding with the lambda calculus. In *Proceedings of IJCAI*, Milan, Italy, 1987.
- [Litman and Devanbu, 1990] D. Litman and P. Devanbu. Clasp (classification of scenarios and plans): User's guide. Technical report, Bell Laboratories, 1990.
- [MacGregor, 1988] R. M. MacGregor. A deductive pattern matcher. In *Proceedings of AAAI*, pages 403–408, St. Paul, MN, 1988.
- [Patel-Schneider and Borgida, 1991] P.F. Patel-Schneider and A. Borgida. The (non-standard) treatment of individuals and user functions in classic concept definitions. Technical report, Bell Laboratories, 1991.
- [Rich, 1981] C. Rich. A formal representation of plans in the programmer's apprentice. In *Proceedings, 7th International Joint Conference on Artificial Intelligence*, Vancouver, Canada, August 1981.
- [Sacerdoti, 1977] E. D. Sacerdoti. *A Structure for Plans and Behavior*. Elsevier, New York, 1977.
- [Sedgewick, 1988] R. Sedgewick. *Algorithms*. Addison Wesley, 1988.
- [Sidner, 1985] Candace L. Sidner. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1(1):1–10, February 1985.
- [Soloway and Ehrlich, 1984] E. Soloway and K. Ehrlich. Empirical studies of programming knowledge. *IEEE Transactions on Software Engineering*, 10(5), September 1984.
- [Soloway *et al.*, 1988] E. Soloway, J. Pinto, S. Letovsky, D. Littman, and R. Lampert. Designing documentation to compensate for de-localized plans. *Communications of the ACM*, 31(11), November 1988.
- [Swartout and Neches, 1986] W. Swartout and R. Neches. The shifting terminological space: An impediment to evolvability. In *Proceedings of AAAI*, pages 936–941, 1986.
- [Tenenberg, 1989] J. D. Tenenberg. Inheritance in automated planning. In *Proceedings First International Conference on Principles of Knowledge Representation and Reasoning*, pages 475–485, Toronto, Ontario, Canada, 1989.
- [Wellman, 1988] M. P. Wellman. *Formulation of Tradeoffs in Planning Under Uncertainty*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, August 1988.
- [Yen *et al.*, 1989] J. Yen, R. Neches, and R. MacGregor. Using terminological models to enhance the rule-based paradigm. In *Proceedings of Second International Symposium on Artificial Intelligence*, October 1989.

A Formal Basis For Analogical Reasoning

Charles Dierbach

Dept. of Computer and Information Sciences
University of Delaware
Newark, DE 19716
dierbach@udel.edu

Daniel L. Chester

Dept. of Computer and Information Sciences
University of Delaware
Newark, DE 19716
chester@udel.edu

Abstract

We present here a mathematical model providing a formal basis for analogical reasoning, referred to as the Abstractional Concept Mapping Theory. The approach provides a formal basis for all comparison-based reasoning (e.g., literal similarity, analogy, metaphor, and scientific models) in that given any two descriptions in first-order logic, arbitrary comparisons of similarity are representable as first-order theories (referred to as "comparison theories"), dependent only on the notion of logical truth, and not on domain-specific heuristics or particular features of the knowledge representation.

1 INTRODUCTION

In this work, we view analogical reasoning as a fundamental cognitive process and an essential component to all comparison-based reasoning, e.g., literal similarity,¹ analogy, metaphor, and scientific models. (For a comprehensive discussion of the role of analogical reasoning in these and other aspects of cognition, see [Indurkhy 1986].) Analogical reasoning is seen both as a *matching process* in determining correspondences between aspects of two descriptions, and as an *inference method* in inferring aspects of one description based on aspects of the other. We refer to these processes as "analogical matching" and "analogical inference," respectively.

Given that the process of analogical matching necessarily precedes the process of analogical inference, this research addresses the question of how comparisons between two given descriptions can be discovered, represented, and used as a basis for analogical inference in intelligent systems. Two main beliefs be-

hind our approach are the following: (a) the focus of such comparisons should be semantically based, rather than syntactically based; and (b) in order to represent comparisons in general, an explicit representation of an abstraction common to the two descriptions is required.

We present here a theory of analogical reasoning referred to as the Abstractional Concept Mapping Theory. The approach provides a general theory in that given any two descriptions in first-order logic, arbitrary comparisons of the descriptions are representable as first-order theories (referred to as "comparison theories"), dependent only on the notion of logical truth, and not on domain-specific heuristics or particular features of the knowledge representation. Thus, the Abstractional Concept Mapping Theory provides a model for the explicit representation of analogical comparisons, seen as prerequisite to the process of analogical inference.

2 RELATED WORK

Most computational theories of analogical reasoning are domain-specific or knowledge-representation sensitive (see [Winston 1980], [Araya 1983], [Burstein 1983], [Carbonell 1983], [Douglas and Moran 1983], [Falkenhainer 1987], [Greiner 1988], [Kedar-Cabelli 1988], [Shinn 1988]), the focus being restricted to problem solving or learning. Little work, however, has been done in the development of general, domain-independent computational models of analogical reasoning. Three exceptions to this are the Structure-Mapping Theory of Gentner [1983, 1988], the Theory of Constrained Semantic Transference of Indurkhy [1985, 1986a, 1988], and the use of "determination rules" by Davies and Russell [1987]. Each provides a domain-independent, general model of analogical inference that incorporates or presupposes a model of analogical matching. We briefly discuss the work of Gentner and Indurkhy.

¹We consider complete literal similarity as "degenerate analogy," and thus see such comparisons as involving analogical reasoning.

```

SOLAR SYSTEM DOMAIN
ATTRACTS(sun,planeti)
MORE_MASSIVE_THAN(sun,planeti)
REVOLVES_AROUND(planeti,sun)
HOTTER_THAN(sun,planeti)
YELLOW(sun)
CAUSES(ATTRACTS(sun,planeti) ∧
        MORE_MASSIVE_THAN(sun,planeti),
        REVOLVES_AROUND(planeti,sun))

ATOM DOMAIN
ATTRACTS(nucleus,electroni)
MORE_MASSIVE_THAN(nucleus,electroni)
REVOLVES_AROUND(electroni,nucleus)

```

Figure 1: Solar System and Atom Domains.

2.1 GENTNER'S STRUCTURE-MAPPING THEORY

2.1.1 Overview

Gentner's Structure-Mapping Theory is a general, domain-independent model of analogical inference within the framework of a propositional network knowledge representation, i.e., it attempts to determine "mappable structures" within the knowledge representation of one description that can be transferred to a second description in a way that corresponds to a psychologically plausible instance of analogy. The key ideas to the mapping process are: (a) attributes are dropped from (are incidental to) the mapping,² and (b) which relations are mapped is determined strictly syntactically by what Gentner calls the "Systematicity Principle," which states that those relations most likely mapped as part of the analogy relation are those most highly interconnected to other relations, as determined by the existence of higher-order relations.

We take the solar system / atom analogy example from [Gentner 1983], given in Figure 1. Assuming object correspondences sun ↔ nucleus and planet_i ↔ electron_i, we find an instance of sound analogical inference by the Structure-Mapping Theory to be represented by the introduction of the following structure into the atom domain, as dictated by the Systematicity Principle:

```

CAUSES(ATTRACTS(nucleus,electroni) ∧
       MORE_MASSIVE_THAN(nucleus,electroni),
       REVOLVES_AROUND(electroni,nucleus)).

```

²Gentner proposes a continuum between literal similarity and analogy, whereby the more attributes vs. relations mapped, the more the comparison is seen as one of literal similarity, and the more relations vs. attributes mapped, the more the comparison is seen as analogy. Mapping of attributes may occur, therefore, but are taken as separate from the analogy relation itself.

2.1.2 Criticism

Although we agree that the Systematicity Principle can serve as a means of identifying sound analogical inference, and implementation of the theory has served useful in gaining insight into the psychological aspects of analogical reasoning (see [Falkenhainer, Forbus and Gentner 1989]), the main criticism we have of the Structure-Mapping Theory as a general computational model is that it is too syntactically based.³ We summarize our specific points of criticism below:

- The distinction of "attribute" and "relation," whereby attributes are taken as one-argument predicates and relations as predicates of two or more arguments, presupposes such a representation of these aspects in a given description.⁴
- The reliance of the Systematicity Principle on the existence of higher-order relations in identifying "mutually-interconnecting relationships" assumes the existence of such higher-order relations in identifying "mappable structures."
- The requirement that predicates of the source domain map to identical predicates of the target domain assumes a common set of predicates in each description.⁵
- Inflexibility in the mapping of terms (i.e., individuals map to individuals and function terms to function terms, but not arbitrary terms to arbitrary terms) potentially prevents objects of each description from being placed into correspondence.

It should be understood that the source of our criticism of the Structure-Mapping Theory comes from the different perspective we have on the problem of analogical reasoning; whereas Gentner has developed a model in which such syntactic restrictions has allowed for the implementation of a computationally feasible theory for the exploration of the psychological process of analogy, we instead focus on the development of a general computational model that is insensitive to such syntactic restrictions.

2.2 INDURKHYA'S THEORY OF CONSTRAINED SEMANTIC TRANSFERENCE

2.2.1 Overview

Indurkhyā's Theory of Constrained Semantic Transference, although presented as a general, domain-

³Gentner herself addresses the knowledge representation sensitivity of the theory by stating that "These representations ... are intended to reflect the way people construe a situation rather than what is logically possible." [Gentner 1983, p. 157].

⁴See [Gentner 1983, footnote 3] for a discussion.

⁵The requirement that predicates map as identity, however, often cited as overly restrictive (see [Burstein 1983], [Indurkhyā 1985], [Shinn 1988]), apparently is being reconsidered by Gentner – whereas in an earlier paper Gentner defended such an identity mapping as necessary (see [Gentner 1983, footnote 6]), in a more recent paper we find a reconsideration of this syntactic constraint (see [Falkenhainer, Forbus and Gentner 1989]).

FAMILY DOMAIN
$\forall z \forall y [\text{FATHER}(z, y) \Rightarrow \forall z [\text{FATHER}(z, y) \Leftrightarrow z = x]]$
"A father of any given person is unique."
etc.
TREE DOMAIN
$\forall z \forall y [\text{ARC}(z, y) \Rightarrow \forall z [\text{ARC}(z, y) \Leftrightarrow z = x]]$
"A parent node of any given node is unique."
etc.

Figure 2: Family and Tree Domains.

independent theory of both metaphor and analogy within the framework of a variation of first-order logic, is most well-developed as a theory of metaphor. The primary advantage of Indurkhya's theory over Gentner's as a general computational model of analogical reasoning is that Indurkhya's mapping is more of a semantic mapping, basing such mappings on the notion of logical truth, as opposed to the identical mapping of predicates that Gentner's theory requires.

The basic assumption of Indurkhya's theory is that "in metaphor, or analogy, some terms of the source domain are applied to terms of the target domain." [Indurkhya 1985, p. viii]. A T-MAP, therefore, is defined as a mapping of the vocabulary of the source domain to the vocabulary of the target domain, applied to a given set of "structural constraints" (sentences). T-MAPS which introduce into the target domain statements that are new to (but not inconsistent with) the target domain are referred to as *weakly coherent*. Such mappings (along with some additional special operators) are seen as characterizing interpretations of metaphor, and are the central focus of Indurkhya's theory. T-MAPS which introduce into the target domain statements that are already true of the target domain are referred to as *strongly coherent*, and are seen as characterizing interpretations of analogy. As an example of statements put into correspondence by a strongly coherent T-MAP, we give the example in Figure 2 from [Indurkhya 1985].

2.2.2 Criticism

Although we see Indurkhya's theory as a step towards a more semantic mapping than that provided by Gentner's approach, we see problems with the use of T-MAPS as the basis for a general computational model of analogical reasoning. We summarize our specific points of criticism below:

- No distinction is made in the mapping of attributes versus the mapping of relations. Given the notion of analogy as corresponding *structure* between descriptions, we see this distinction as necessary.
- Given the symbol-to-symbol map of T-MAPS, corresponding statements must have the same syntactic form. Therefore, we still find the theory to be syntactically constrained.
- T-MAPS involve partial *functions* between vocabularies, thereby constituting a direct mapping.

We elaborate on the last point above with an example. Consider the four-part geometric analogy problem in Figure 3. One analogy relation here can be captured by the mapping of statements S_1, S_2, S_6 and S_7 . The analogy captured by these statements can be stated as "Three boxes spatially related at time t_1 , then reverse spatially related at time t_2 ." However, the complete analogy here includes the fact that "the boxes are to the left of a larger box at time t_1 , and inside of that box at time t_2 ." This extra information to be captured is in statements S_3, S_4, S_5, S_8, S_9 and S_{10} . In order to make these statements transformable by the existing mapping, however, it is required that *INSIDE_OF* map *both* to *LEFT_OF* and *INSIDE_OF*. Thus, a function mapping of the vocabulary is too restrictive here; a more general mapping is required.

As a result of the above, we find the characterization of analogy as strongly coherent T-MAPS on the one hand as inappropriately capturing some comparisons as analogy due to the arbitrary mapping of attributes, and on the other hand as unable to capture some comparisons normally seen as analogy, due to the direct function mapping provided by a T-MAP (as shown in Figure 3). We therefore see the need for a more clear separation between mappings identifying analogical interpretation versus those providing metaphorical interpretation.

3 THE ABSTRACTIONAL CONCEPT MAPPING THEORY

We present a model of analogical matching within the framework of first-order logic that allows for a true semantic mapping between two descriptions; that is, the mapping process is dependent only on the notion of logical truth, and not on the particular vocabulary used within each description or the syntactic structure of individual statements. We see such a model of analogical matching as necessary before a sufficiently general model of analogical inference can be developed.

Given any two descriptions in first-order logic, therefore, the Abstractional Concept Mapping Theory provides a means for the explicit representation of abstractions common to both descriptions; these abstractions are characterized by first-order theories of logic, referred to as "comparison theories." We first discuss the representation of descriptions within our theory, then describe the mapping process underlying comparisons of such descriptions, and finally give characterizations of comparisons identifying analogical and literal similarity.

3.1 REPRESENTATION OF BACKGROUND AND SITUATION-SPECIFIC KNOWLEDGE

We introduce a knowledge representation within first-order logic based on a pragmatic distinction of "situation-specific" knowledge and background knowledge. The pragmatic separation of one's background knowledge of the world (e.g., the meaning

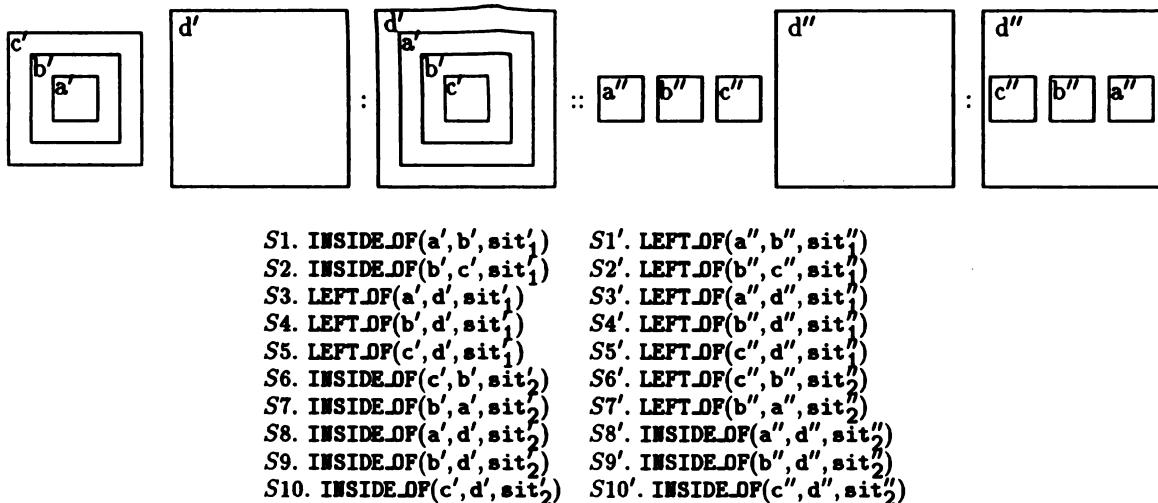


Figure 3: A Four-Part Geometric Analogy Problem.

of LEFT_OF) from situation-specific knowledge (e.g., LEFT_OF(a,b)) enables a clear distinction between corresponding *equivalent* statements, providing identification of literal similarity between two given descriptions, and corresponding *isomorphic* statements, providing identification of analogical similarity.

3.1.1 “Definite” vs. “Indefinite” Constants

Universal L-vocabulary, \mathcal{L}

We introduce the “universal set of meaning formulas,” \mathcal{M} , to be an arbitrary (non-empty) finite set of sentences over the “universal L-vocabulary,” \mathcal{L} , such that \mathcal{M} is consistent. The set \mathcal{M} is intended to represent all of a given individual’s background knowledge of the world,⁶ e.g.,

$$\{\forall x \forall y \forall z [[\text{LEFT_OF}(x, y) \wedge \text{LEFT_OF}(y, z)] \Rightarrow \text{LEFT_OF}(x, z)], \text{HAS_CAPITOL}(\text{Delaware}, \text{Dover}), \dots\}.$$

We refer to each constant of \mathcal{L} as a “definite constant,” given that an intended interpretation of such a constant is given (constrained) by \mathcal{M} .

Universal A-vocabulary, \mathcal{A}

Given that temporary names are often introduced in situation-specific knowledge to identify objects and concepts not provided by the background knowledge, to provide such names (which have indefinite meaning in that they are interpreted differently in different situations), we introduce the “universal A-vocabulary,” \mathcal{A} , to be an arbitrarily large vocabulary such that no constant of \mathcal{A} appears in \mathcal{L} . The only set of symbols shared by the two vocabularies is an arbitrarily large

⁶The requirement that \mathcal{M} be *consistent* is an idealized approach, in that individuals often have inconsistent knowledge of the world. We see this, however, as a separate issue from the issues being addressed in this work.

set of individual variables, \mathcal{V} , referred to as the “universal set of individual variables.” We refer to each constant of \mathcal{A} as an “indefinite constant,” since the interpretation of such symbols is not constrained by the set of sentences, \mathcal{M} . In order to distinguish the constant symbols of \mathcal{A} from the constant symbols of \mathcal{L} , we denote indefinite constants by symbols of length one (e.g., a to denote a particular object, or P_1 a particular predicate), and definite constants by symbols of length two or more (e.g., Delaware , LEFT_OF).

Three General Vocabulary Types

Having defined the two universal vocabularies given above, we define three general types of vocabularies relative to \mathcal{L} and \mathcal{A} . An “L-vocabulary” is intended to be used in describing situation-specific knowledge. Thus, definite constants such as LEFT_OF and Delaware , whose meaning are given by the background knowledge in \mathcal{M} , are used in such descriptions. Given, however, that situation-specific knowledge generally involves unique objects that do not appear in \mathcal{M} (such as a particular stack of boxes), we allow an arbitrary number of indefinite individual (and function) constants in an L-vocabulary for denoting such objects, such as a, b in $\text{ON_TOP_OF}(a, b)$. An “A-vocabulary,” on the other hand, is used to represent knowledge abstracted from two given situations, and thus contains only indefinite constants. Lastly, a “G-vocabulary” (general vocabulary) is a vocabulary containing any number of definite and indefinite constants. Thus, L-vocabularies and A-vocabularies are special cases of G-vocabularies. We refer to sentences in general over any one of these vocabulary types, therefore, as a “G-formula.”

3.1.2 Structural Descriptions

In order to formally represent a description or situation, we define a “structural description” to be a three-tuple, $D = \langle S, G, K \rangle$, where S is a set of sentences over

$$D_1 = \langle S_1, G_1, K_1 \rangle$$

$$\begin{aligned} S_1 = & \{ \text{BOX}(a), \text{BOX}(b), \text{BOX}(c), \\ & \exists x[\text{TABLE}(x) \wedge \text{ON_TOP_OF}(c, x)], \\ & \text{ON_TOP_OF}(a, b), \text{ON_TOP_OF}(b, c), \\ & \forall x[\text{BOX}(x) \Rightarrow \text{RED}(x)] \}, \end{aligned}$$

$$G_1 = \langle \{a, b, c\}, \{ \}, \{\text{BOX}, \text{TABLE}, \text{ON_TOP_OF}, \text{RED}\}, \mathcal{V} \rangle,$$

$$\begin{aligned} K_1 = & \{ \forall x[\text{BOX}(x) \Rightarrow \text{OBJECT}(x)], \\ & \forall x \forall y[\text{ON_TOP_OF}(x, y) \Rightarrow \neg \text{ON_TOP_OF}(y, x)], \\ & \forall x \forall y \forall z[(\text{ON_TOP_OF}(x, y) \wedge \text{ON_TOP_OF}(y, z)) \Rightarrow \\ & \quad \text{ABOVE}(x, z)], \\ & \forall x \forall y[\text{ABOVE}(x, y) \equiv \text{BELOW}(y, x)], \\ & \text{etc.} \}. \end{aligned}$$

Figure 4: L-Based Structural Description, D_1 .

$$D_2 = \langle S_2, G_2, K_2 \rangle$$

$$\begin{aligned} S_2 = & \{ p_1(a), p_1(b), p_1(c), \\ & p_2(a, b), p_2(b, c), \\ & \forall x \forall y[p_2(x, y) \Rightarrow \neg p_2(y, x)], \\ & \forall x \forall y[(p_2(x, y) \wedge p_2(y, z)) \Rightarrow p_2(x, z)] \}, \end{aligned}$$

$$G_2 = \langle \{a, b, c\}, \{ \}, \{p_1, p_2\}, \mathcal{V} \rangle,$$

$$K_2 = \{ \}.$$

Figure 5: A-Based Structural Description, D_2 .

a G-vocabulary, G , and K is a subset of \mathcal{M} such that $S \cup K$ is consistent.⁷ We call S the “initial description” of the structural description, intended to be a “natural” representation of the description, and refer to the set K as a “K-domain” (knowledge domain), taken as all the background knowledge of \mathcal{M} needed for the interpretation of S .⁸ The intention, here, is that K represents an identifiable body of knowledge within \mathcal{M} needed for the interpretation of S , e.g., “spatial relations,” “world politics.” We give in Figure 4 an example of a structural description in which G_1 is an L-vocabulary (i.e., containing only predicate constants that are definite constants), hence referred to as an “L-based structural description,” and in Figure 5 a structural description in which G_2 is an A-vocabulary (i.e., containing only indefinite constants), hence referred to as an “A-based structural description.”

⁷We see the requirement that $S \cup K$ be consistent as necessary in order that structural descriptions be interpretable in the world represented by \mathcal{M} – the universal set of meaning formulas.

⁸We conclude that given the inherent nature of a K-domain as the relevant background knowledge of a given description, such a K-domain cannot be extracted from \mathcal{M} by formal means given that the notion of “relevance” cannot be captured formally. We therefore resort to assuming that the background knowledge needed for a particular description is predetermined and explicitly included with a given structural description.

$$\begin{aligned} \dot{S}_1 = & \{ \text{DOMAIN_OBJ}(a), \text{DOMAIN_OBJ}(b), \text{DOMAIN_OBJ}(c), \\ & \text{DOMAIN_OBJ}(f_1), \\ & \text{BOX}(a), \text{BOX}(b), \text{BOX}(c), \\ & \text{TABLE}(f_1) \wedge \text{ON_TOP_OF}(c, f_1), \\ & \forall x[\text{DOMAIN_OBJ}(x) \Rightarrow [\text{BOX}(x) \Rightarrow \text{RED}(x)]] \}. \end{aligned}$$

Figure 6: Normalized Initial Description of D_1 .

$$\begin{aligned} \dot{S}_2 = & \{ \text{DOMAIN_OBJ}(a), \text{DOMAIN_OBJ}(b), \text{DOMAIN_OBJ}(c), \\ & p_1(a), p_1(b), p_1(c), p_2(a, b), p_2(b, c), \\ & \forall x \forall y[(\text{DOMAIN_OBJ}(x) \wedge \text{DOMAIN_OBJ}(y)) \Rightarrow \\ & \quad [p_2(x, y) \Rightarrow \neg p_2(y, x)]], \\ & \forall x \forall y \forall z[(\text{DOMAIN_OBJ}(x) \wedge \text{DOMAIN_OBJ}(y) \wedge \\ & \quad \text{DOMAIN_OBJ}(z)) \Rightarrow \\ & \quad [[p_2(x, y) \wedge p_2(y, z)] \Rightarrow p_2(x, z)]] \}. \end{aligned}$$

Figure 7: Normalized Initial Description of D_2 .

3.1.3 Normalized Initial Descriptions

Having introduced the definition of a structural description, we now introduce the notion of a “normalized initial description.” Such a translation of an initial description has two purposes – first, to have each entity of the description denoted by a definite term; and second, to declare all such entities as “domain objects,” i.e., as specifically part of the situation described.

We want each entity to be represented by a unique definite term because in the eventual mapping between two descriptions, we need to construct object correspondences. Existentially quantified statements, such as $\exists x[\text{BOX}(x) \wedge \text{RED}(x)]$, are not guaranteed to be satisfied by existing definite terms. We therefore make use of Skolemization for obtaining definite terms from such statements, e.g., $\exists x[\text{BOX}(x) \wedge \text{RED}(x)]$ can be Skolemized to $[\text{BOX}(f_1) \wedge \text{RED}(f_1)]$, where the Skolem function f_1 provides a definite term for the eventual object mapping.

The reason for declaring each definite term of an initial description as a “domain object” is to allow intensional definitions of property and relation that do not depend strictly on their syntactic form, as follows. We assume that \mathcal{L} contains the predicate DOMAIN_OBJ , used to distinguish the set of objects seen as part of a structural description, D . We call this distinguished set of objects the “domain objects of D .” We further assume that all entities mentioned in an initial description of a structural description, and no others, are the intended domain objects (as distinguished from entities in the background knowledge of \mathcal{M}), and thus introduce statements of the form $\text{DOMAIN_OBJ}(t)$ for each definite term t appearing in the initial description after Skolemization. Briefly, then, statements made on two or more domain objects are taken as “relations,” and statements involving only one domain object are taken as “properties.” We point out the special treatment of universally quantified formulas of a given initial description when normalized in that wffs of the form $\forall x P(x)$ are converted to the form $\forall x[\text{DOMAIN_OBJ}(x) \Rightarrow P(x)]$ to restrict quantification

over the objects seen as part of the description, and not over objects introduced with the associated K-domain (i.e., background knowledge) of the structural description.

We denote the normalized initial description of a structural description $D = \langle S, G, K \rangle$ by \dot{S} , and give in Figure 6 and Figure 7 the normalized initial descriptions of structural descriptions D_1 and D_2 above. For a complete discussion of the normalization process, see [Dierbach 1991].

3.1.4 Descriptive Closure

All statements that can be inferred about a particular situation as described by a given structural description need to be made explicit if all correspondences that can be made between a pair of structural descriptions are to be found. This is done by applying the associated K-domain of a structural description to its initial description. For example, if one were presented with the statements $\text{LEFT_OF}(a, b)$ and $\text{LEFT_OF}(b, c)$, one could infer other statements such as $\text{LEFT_OF}(a, c)$ given the common understanding of the meaning of LEFT_OF . However, in first-order logic, the above statements of $\text{LEFT_OF}(a, b)$, $\text{LEFT_OF}(b, c)$ are no more constrained in interpretation than the statements $P_1(a, b), P_1(b, c)$. It is the choice of predicate constant LEFT_OF over P_1 that causes us to impart more knowledge to the statements than formally represented.

Given our differentiation of definite versus indefinite constants, we formally capture the above informal distinction by defining the “descriptive closure” of a given structural description to be the logical closure of the statements of the (normalized) initial description union the statements of the associated K-domain. That is, for a structural description $D = \langle S, G, K \rangle$, we define the descriptive closure of D , denoted by D^+ , to be the logical closure of $\dot{S} \cup K$, and refer to the vocabulary of D^+ as the “domain vocabulary” of D , denoted D^c .

We point out that by this definition, D^+ may contain statements not directly related to the situation described. If one or more of the definite terms of a G-formula is declared as a domain object, as in the statement $\text{ABOVE}(a, c)$, then we refer to that statement as a “descriptive statement,” given that it makes a statement involving at least one object of the structural description, and thus a statement particular to the situation.⁹ We also consider wffs of the form $\forall z_1 \dots \forall z_n [[\text{DOMAIN_OBJ}(z_1) \wedge \dots \wedge \text{DOMAIN_OBJ}(z_n)] \Rightarrow \phi]$

⁹In this example, all objects of the descriptive closure of D_1 are declared as domain objects, and thus all ground statements in D_1^+ are descriptive statements (of the situation). However, this is not always the case. New objects may be introduced by the background knowledge in the associated K-domain. For example, in a structural description of the solar system in which the only objects explicitly mentioned in the initial description are sun, earth, planet, we may have introduced by the background knowledge “non-domain” object alpha.centuri.

$$\begin{aligned} D_1^+ = & \{\text{DOMAIN_OBJ}(a), \text{DOMAIN_OBJ}(b), \\ & \text{DOMAIN_OBJ}(c), \text{DOMAIN_OBJ}(f_1), \\ & \text{BOX}(a), \text{BOX}(b), \text{BOX}(c), \\ & \text{OBJECT}(a), \text{OBJECT}(b), \text{OBJECT}(c), \\ & \text{TABLE}(f_1) \wedge \text{ON_TOP_OF}(c, f_1), \\ & \text{ON_TOP_OF}(a, b), \text{ON_TOP_OF}(b, c), \\ & \sim \text{ON_TOP_OF}(b, a), \sim \text{ON_TOP_OF}(c, b), \\ & \text{ABOVE}(a, c), \text{BELOW}(c, a), \\ & \text{etc.} \\ & \forall z [\text{DOMAIN_OBJ}(z) \Rightarrow [\text{BOX}(z) \Rightarrow \text{RED}(z)]], \\ & \forall z [\text{DOMAIN_OBJ}(z) \Rightarrow [\text{BOX}(z) \Rightarrow \text{OBJECT}(z)]], \\ & \forall z \forall y [\text{ON_TOP_OF}(z, y) \Rightarrow \sim \text{ON_TOP_OF}(y, z)], \\ & \forall z \forall y [[\text{DOMAIN_OBJ}(z) \wedge \text{DOMAIN_OBJ}(y)] \Rightarrow \\ & \quad [\text{ON_TOP_OF}(x, y) \Rightarrow \sim \text{ON_TOP_OF}(y, x)]], \\ & \text{etc. } \} \end{aligned}$$

Figure 8: Descriptive Closure of D_1 .

as “domain-related meaning formulas,” and thus also related to the situation described. In general, we refer to descriptive statements and domain-related meaning formulas as “domain-related G-formulas,” and it is only such wffs in D^+ that we consider as specific to the situation. We give the descriptive closure of structural description D_1 in Figure 8.

3.1.5 Concept Spaces

Because the purpose of a description comparison is to allow the representation of a comparison between two (structural) descriptions that does not depend on the choice of vocabulary within each or the syntactic structure of individual statements, we ultimately provide an indirect mapping that allows a pairing of “concepts” with “concepts,” rather than a pairing of predicates with predicates. We do this by defining the “concept space” over a given structural description D , denoted by D° , to be all the lambda expressions constructible over the vocabulary of the descriptive closure of the structural description (i.e., “domain vocabulary” of D). Since lambda expressions are not statements, but derive statements when instantiated with a particular set of terms, we see lambda expressions as a suitable form for the representation of concepts. We give examples of such concepts in Figure 9.

3.2 THE MAPPING PROCESS

In our theory, we provide a means of putting into correspondence aspects of two descriptions by construction of a third “abstract” structural description (referred to as a “comparison-based description”), and a pair of mappings such that each aspect of the comparison-based description gets mapped into two aspects, one from each “component description” of the comparison, as shown in Figure 10. We adopt the point of view, therefore, that the explicit representation of an abstraction of two descriptions is necessary in order to allow arbitrary analogical comparisons of the two.¹⁰

¹⁰Discussion of the need for the explicit representation of an abstraction (often called “schema”) common to two de-

Concepts

1. $[\lambda(x_1, x_2)\text{LEFT_OF}(x_1, x_2)]$.
This lambda expression denotes the concept of one object being left of another. If G included the predicate RIGHT_OF , G° would also contain $[\lambda(x_1, x_2)\text{RIGHT_OF}(x_2, x_1)]$. Thus, there may exist equivalent concepts in any given concept space.
2. $[\lambda(x_1)\text{LEFT_OF}(x_1, x_1)]$.
This lambda expression denotes the concept of an object being left of itself. Thus, a given concept may have no instantiations consistent with the universal set of meaning formulas, \mathcal{M} .
3. $[\lambda(x_1, \dots, x_n)\text{ON}(x_1, x_2) \wedge \dots \wedge \text{ON}(x_{n-1}, x_n)]$.
This lambda expression demonstrates how for any given vocabulary, an infinite number of concepts can be represented (i.e., the concept of *two* stacked objects, *three* stacked objects, etc.).
4. $[\lambda(x_1, x_2)p_1(x_1, x_2) \wedge \text{LEFT_OF}(x_1, x_2)]$.
This lambda expression demonstrates that a given concept may be *indefinite* in that the corresponding lambda expression may contain indefinite constants, as indicated here by p_1 .
5. $[\lambda(x_1)\text{CHAIR}(x_1) \wedge \exists y[\text{CHAIR}(y) \wedge \text{NEXT_TO}(y, x_1)]]$.
This lambda expression corresponds to the concept of a chair that has another chair next to it, thus demonstrating that a given concept may contain existentially-quantified individual variables.
6. $[\lambda(x_1)\text{CHAIR}(x_1) \wedge \forall y[\text{CHAIR}(y) \Rightarrow \text{SMALLER_THAN}(x_1, y)]]$.
This lambda expression corresponds to the concept of a smallest chair, thus demonstrating that a given concept may contain universally-quantified individual variables.
7. $[\lambda() \exists z[\text{BOX}(z) \wedge \text{RED}(z)]]$.
This lambda expression demonstrates that a concept may be of order 0.

Figure 9: Examples of Concepts.

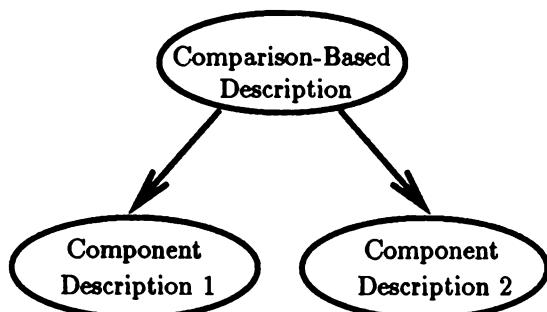


Figure 10: Indirect Mapping.

However, in order that such mappings be semantically based, we propose in our model a definition of an analogical mapping such that *concepts*, represented by lambda expressions, are paired with *concepts*, rather than mapping predicate symbols to predicate symbols. Thus, we define an analogical mapping as an indirect mapping that puts into correspondence concepts from each component description's "concept space," along with a correspondence of terms (objects). The intuition behind this approach is that for a given pair of descriptions, corresponding aspects should be mappable independently of the syntactic structure of individual statements or vocabulary used. Thus, for statements $\text{BETWEEN}(b', a', c') \wedge \text{LEFT_OF}(a', c')$ and $\text{RIGHT_OF}(c'', b'') \wedge \text{RIGHT_OF}(b'', a'')$, a simple syntactic mapping of terms to terms and predicate symbols to predicate symbols would not suffice here, for example, given that the two statements do not have the same syntactic form. We characterize in our theory, therefore, the pairing of such statements based strictly on the notion of logical truth (i.e., with the appropriate background knowledge, it can be *derived* that the above statements are equivalent, given the appropriate object correspondences.)

We therefore, in our analogical mapping, pair equivalent (isomorphic) concepts between the concept spaces of two descriptions. Equivalent (isomorphic) concepts having instantiations within the respective descriptions are seen as identifying equivalent (isomorphic) *statements* between the descriptions, thus identifying an instance of literal similarity (analogy). Isomorphic concepts themselves are seen as identifying a metaphorical mapping between two domains of knowledge – that is, identifying analogical concepts across two domains of knowledge.¹¹ We give below a more formal discussion of the mapping process.

3.2.1 Term Mappings

We define an "object mapping" to be a one-to-one mapping of individual constants of one G -vocabulary to definite terms over another. A pair of such object mappings ultimately puts into a one-to-one correspondence a particular set of objects of each description. Since a comparison-based description is an abstraction of what is common to both component descriptions, we allow an object mapping to be a mapping from individual constants to *arbitrary* definite terms such that arbitrary objects (denoted by such terms) can be put into correspondence, such as

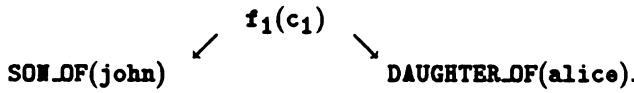
$$\text{SOM_OF(john)} \quad c_1 \quad \text{helen.}$$

In this case, given that no corresponding function constant to SOM_OF exists in the term helen , we see no usefulness in mapping a corresponding function constant

descriptions in analogical reasoning can be found in [Gick and Holyoak 1983], [Carbonell and Minton 1985], and [Shinn 1988].

¹¹See [Tourangeau and Sternberg 1982] for a discussion of metaphor as analogous concepts across two domains of knowledge.

of the comparison-based description to the function constant **SON_OF**. However, in placing into correspondence definite terms both of which are function terms, we wish to capture the structure of the terms in the comparison-based description, such as



Thus, we define a "function mapping" to be a one-to-one mapping of the function constants of one G-vocabulary, to function constants of another. We define a "term mapping," then, to be defined by a given object mapping and a given function mapping.

3.2.2 Definition Mappings

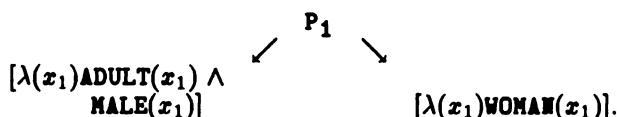
We define a "definition mapping" to be a mapping of the set of predicates of one G-vocabulary to a set of lambda expressions of the concept space over a second G-vocabulary. We view such a mapping as a mapping of predicate constants to "definitions" of each predicate over a second vocabulary. For example, we may have a mapping of an indefinite predicate constant P_1 in a G-vocabulary, G , to the concept space over a G-vocabulary, G_1 , as follows:

$$P_1 \longrightarrow [\lambda(x_1)\text{ADULT}(x_1) \wedge \text{MALE}(x_1)],$$

i.e., such that the "definition" of P_1 by this mapping is "an adult male." If a second definition mapping were defined from P_1 in vocabulary, G , to the concept space over a G-vocabulary, G_2 , we may have:

$$P_1 \longrightarrow [\lambda(x_1)\text{WOMAN}(x_1)].$$

Thus, if G were the vocabulary of a given comparison-based description such that G_1 and G_2 were the domain vocabularies of the component descriptions of the comparison, we have the pairing of concepts $[\lambda(x_1)\text{ADULT}(x_1) \wedge \text{MALE}(x_1)]$ and $[\lambda(x_1)\text{WOMAN}(x_1)]$ as follows:



3.2.3 Formula Mappings and A-MAPS

Given the above mappings for the mapping of predicate constants and terms, we give a "formula mapping" to be defined by a given term mapping and a given definition mapping, allowing wffs over one G-vocabulary to be mapped to wffs over another. The most distinctive feature of such a formula mapping is that arbitrary wffs may be placed into correspondence between two component descriptions of a given descriptions comparison, by virtue of the flexibility offered by a definition mapping. We do this by defining a pair of formula mappings from a common G-vocabulary as shown in Figure 11. We refer to such a "split" formula mapping as an "A-MAP." Such a mapping allows arbitrary wffs to be put into correspondence. Formally, an A-MAP, $\hat{T} = \langle G, G_1, G_2, T_1, T_2 \rangle$,

is a set of G-vocabularies G, G_1, G_2 and formula mappings T_1 and T_2 such that T_1 is an \mathcal{L} -consistent formula mapping from G^* (the language over vocabulary G) to G_1^* , and T_2 is an \mathcal{L} -consistent formula mapping from G^* to G_2^* . (The consequence of requiring that T_1 and T_2 be \mathcal{L} -consistent formula mappings is that formulas such as **LEFT_OF(a, b)** map to a formula such as **LEFT_OF(a', b')**, given that **LEFT_OF** is a definite constant.)

3.3 DESCRIPTION COMPARISONS

Comparisons between a given pair of structural descriptions, referred to as "description comparisons," are formally represented by a given comparison-based description and an associated A-MAP that puts into correspondence wffs of one structural description with wffs of the other, as discussed above. As demonstrated in Figure 3 in the discussion of Indurkhya's theory, a *direct* mapping between descriptions does not provide for a flexible enough mapping to represent all comparisons between a given pair of descriptions. Thus, we solve the problem cited there by allowing the comparison-based description to contain indefinite predicate constants such that by the given A-MAP, an indefinite predicate constant P_1 puts into correspondence concepts $[\lambda(x_1, x_2)\text{INSIDE_OF}(x_1, x_2)]$ and $[\lambda(x_1, x_2)\text{LEFT_OF}(x_1, x_2)]$, and P_2 puts into correspondence concepts $[\lambda(x_1, x_2)\text{INSIDE_OF}(x_1, x_2)]$ and $[\lambda(x_1, x_2)\text{INSIDE_OF}(x_1, x_2)]$, allowing for the correspondence of

$$\text{INSIDE_OF}(a', b', \text{sit}'_1) \leftrightarrow \text{LEFT_OF}(a'', b'', \text{sit}''_1)$$

as well as

$$\text{INSIDE_OF}(a', d', \text{sit}'_2) \leftrightarrow \text{INSIDE_OF}(a'', d'', \text{sit}''_2).$$

A description comparison, therefore, denoted by $\mathcal{C} = \langle \hat{D}, D_1, D_2, \hat{T} \rangle$, consists of a (comparison-based) structural description $\hat{D} = \langle S, G, \mathcal{K} \rangle$, a pair of structural descriptions $D_1 = \langle S_1, G_1, \mathcal{K}_1 \rangle$ and $D_2 = \langle S_2, G_2, \mathcal{K}_2 \rangle$, and an A-MAP $\hat{T} = \langle G, D'_1, D'_2, T_1, T_2 \rangle$, where D'_1 (D'_2) is the domain vocabulary of D_1 (D_2) (see section 3.1.4), such that:

1. $\mathcal{K} = \mathcal{K}_1 \cap \mathcal{K}_2$.
2. For any $\phi \in S$, $T_1(\phi) \in D'_1$ and $T_2(\phi) \in D'_2$, and $T_1(\phi)$ and $T_2(\phi)$ are domain-related G-formulas.
3. For any definite term t appearing in some $\phi \in S$, $S \vdash \text{DOMAIN_OBJ}(t)$ iff $T_1(\text{DOMAIN_OBJ}(t)) \in D'_1$ and $T_2(\text{DOMAIN_OBJ}(t)) \in D'_2$.

The reason for defining the K-domain of the comparison-based description to be the intersection of the K-domains of the two component descriptions in (1) above will become apparent in the discussion of \mathcal{L} -substituted comparison-based descriptions and \mathcal{M} -axiomatizations to follow. In (2), given that G-formulas can appear in the descriptive closure of a given structural description that are not directly related to the description (see section 3.1.4), we therefore only put into correspondence G-formulas that are

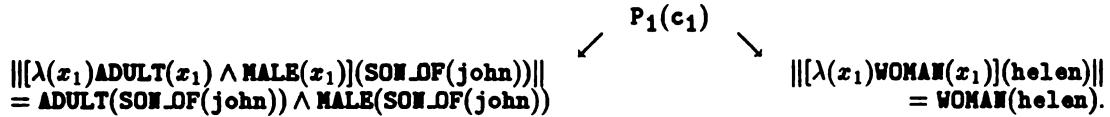


Figure 11: "Split" Formula Mapping.

domain-related G-formulas. Finally, (3) is necessary so that corresponding statements taken as properties (relations) are also taken as properties (relations) when abstracted into the comparison-based description.

3.3.1 Comparison Theories

Given that the comparison-based description of a given description comparison is itself a structural description, the descriptive closure of such a structural description can be taken. Thus, for a description comparison $\mathcal{C} = \langle \hat{D}, D_1, D_2, \hat{T} \rangle$ where $\hat{D} = \langle S, G, K \rangle$, the descriptive closure of \hat{D} , \hat{D}^+ , is defined as the logical closure of $S \cup K$.¹² In the case when K is empty, as is possible when $K_1 \cap K_2$ is empty, the descriptive closure of \hat{D} is equivalent to the logical closure of S . Thus, we see \hat{D}^+ as the explicit representation of all statements common to each component description under the given A-MAP, \hat{T} , and thus refer to \hat{D}^+ as the "comparison theory under \hat{T} " of the description comparison \mathcal{C} above.¹³

3.3.2 Equivalent Concepts and Equivalent Statements

We define "equivalent concepts" to be concepts that can be determined to be equivalent by the background knowledge in \mathcal{M} . Formally, such concepts λ_i, λ_j are considered "equivalent" when λ_i is of the form $[\lambda(x_1, \dots, x_n)\phi_i]$, λ_j is of the form $[\lambda(x_1, \dots, x_n)\phi_j]$ (where ϕ_i and ϕ_j are arbitrary wffs), and $\mathcal{M} \vdash \forall x_1, \dots, \forall x_n [\phi_i \equiv \phi_j]$. For example, it would be expected that the background knowledge would contain meaning formulas such that

$$\mathcal{M} \vdash \forall x \forall y [\text{LEFT_OF}(x, y) \equiv \text{RIGHT_OF}(y, x)],$$

thus making concepts $[\lambda(x_1, x_2)\text{LEFT_OF}(x_1, x_2)]$ and $[\lambda(x_1, x_2)\text{RIGHT_OF}(x_2, x_1)]$ equivalent.¹⁴ Corresponding statements that are instantiations of corresponding equivalent concepts, therefore, are considered equivalent corresponding statements.

¹²We note that comparison-based descriptions need not be normalized, and thus $\hat{D}^+ = S \cup K$ rather than $\hat{D}^+ = \hat{S} \cup K$ here. See [Dierbach 1991] for a discussion.

¹³We have proven in our work that $\forall \phi \in \hat{D}^+, T_1(\phi) \in D_1^+$ and $T_2(\phi) \in D_2^+$.

¹⁴Note that by the above, only "definite concepts" (i.e., lambda expressions containing only definite constants) can be determined to be equivalent, given that \mathcal{M} is a set of sentences over the universal L-vocabulary, \mathcal{L} , which contains only definite constants.

3.3.3 Isomorphic Concepts and Isomorphic Statements

The notion of equivalent concepts was defined by derivability of wffs over the set of meaning formulas in \mathcal{M} . Thus, only "definite concepts," i.e., concepts containing definite constants and no indefinite constants, can be determined as equivalent (given that no indefinite constants appear in the meaning formulas of \mathcal{M}). However, in general, concepts may contain indefinite constants; thus, such corresponding "indefinite concepts" cannot be determined to be equivalent. This is intuitively correct, given that indefinite constants have no fixed, a priori interpretation, and are constrained in interpretation only by the set of wffs in which they occur. In fact, given that indefinite constants are meant to be reused across structural descriptions, the concept $[\lambda(x_1, x_2)\text{P}_1(x_1, x_2)]$ occurring in each concept space of a pair of structural descriptions cannot be determined as equivalent, given that the interpretation of P_1 is dependent only on the set of wffs in which it occurs.

A pair of corresponding concepts, whether definite or indefinite, however, can be determined as isomorphic as follows. First, we introduce the notation $\|\lambda_i(t_1, \dots, t_n)\|$ to represent the wff resulting from the instantiation of lambda expression λ_i with terms t_1, \dots, t_n . We say, then, that for a given A-MAP, \hat{T} , putting into correspondence a set of concepts with concepts and terms with terms across a pair of structural descriptions (say D_1, D_2), concepts $\lambda_i \in D_1^\circ$ (the concept space over D_1) and $\lambda_j \in D_2^\circ$, where λ_i is of the form $[\lambda(x_1, \dots, x_n)\phi_i]$ and λ_j is of the form $[\lambda(x_1, \dots, x_n)\phi_j]$, are "isomorphic concepts under \hat{T} " iff for any terms t_1, \dots, t_n of the comparison-based description,

$$\phi_i(\Gamma_1(t_1), \dots, \Gamma_1(t_n)) \in D_1^+ \text{ iff } \phi_j(\Gamma_2(t_1), \dots, \Gamma_2(t_n)) \in D_2^+,$$

where Γ_1 and Γ_2 are term mappings to the domain vocabularies of D_1 and D_2 , respectively, i.e., such that $t_k \in \{\Gamma_1(t_1), \dots, \Gamma_1(t_n)\}$, $t'_k \in \{\Gamma_2(t_1), \dots, \Gamma_2(t_n)\}$ are arbitrary corresponding definite terms under \hat{T} .

As an example of the above, imagine a description D_1 of three boxes horizontally aligned, and a description D_2 of three boxes vertically aligned. If D_1 had an initial description of $\{\text{LEFT_OF}(a', b'), \text{LEFT_OF}(b', c')\}$ and D_2 an initial description of $\{\text{ABOVE}(a'', b''), \text{ABOVE}(b'', c'')\}$, then for an A-MAP giving the following corresponding expressions

$$\begin{aligned} [\lambda(x_1, x_2)\text{LEFT_OF}(x_1, x_2)] &\leftrightarrow [\lambda(x_1, x_2)\text{ABOVE}(x_1, x_2)], \\ a' &\leftrightarrow a'', \\ b' &\leftrightarrow b'', \\ c' &\leftrightarrow c'', \end{aligned}$$

and no other terms put into correspondence other than a', b', c' and a'', b'', c'' , the fact that for all pairs of corresponding terms t'_i, t''_i and t'_j, t''_j , $\text{LEFT_OF}(t'_i, t'_j)$ is true (of D_1^+) iff $\text{ABOVE}(t''_i, t''_j)$ is true (of D_2^+) we see as formally capturing the notion that the concept of LEFT_OF in one structural description is isomorphic to the concept ABOVE in the other *under the given A-MAP*.¹⁵ (It should be understood that in the above example, even if predicate symbols LEFT_OF and ABOVE were *indefinite* predicate constants, the same definition of isomorphic concepts holds here.)

3.3.4 Description Comparison Types

We specify the types of description comparisons possible used in the eventual characterization of literal and analogical similarity between a pair of structural descriptions. First, we define an “equivalent-statement description comparison” to be a description comparison such that all descriptive statements placed into correspondence are equivalent corresponding descriptive statements. Thus such a description comparison captures something literally similar to both descriptions. An “isomorphic-statement description comparison” is defined to be a description comparison such that all descriptive statements placed into correspondence are isomorphic corresponding descriptive statements. Thus, such a description comparison captures something analogically similar to both descriptions. An “object-abstracted description comparison” is one in which all *relations* put into correspondence are either equivalent or isomorphic corresponding descriptive statements. Given that descriptive statements put into correspondence by a given description comparison may be neither equivalent nor isomorphic, an object-abstracted description comparison ensures that all relations put into correspondence are either equivalent or isomorphic, thereby identifying corresponding structure between two descriptions; properties put into correspondence need not be equivalent nor isomorphic, hence, properties of objects are incidental to the comparison, and in this sense each object is “abstracted” in the comparison. Finally, we define a “total-cover description comparison” to be a description comparison such that for every descriptive statement in the descriptive closure of one component description of the comparison, there exists a corresponding descriptive statement in the other under the associated A-MAP. Thus, a total-cover description comparison places into correspondence every aspect of one description with an aspect of the other.

¹⁵We emphasize that concepts can be determined as isomorphic only relative to a given A-MAP (the same does not hold true for equivalent concepts). This is so because the definition of isomorphic concepts critically depends on a given mapping of terms between two descriptions. In the above example, if the term mapping mapped $a' \leftrightarrow b''$, for instance, or involved the mapping of terms d' and d'' such that $\text{LEFT_OF}(c', d')$ were true and $\text{ABOVE}(c'', d'')$ were not (or vice versa), then the concepts of $[\lambda(x_1, x_2)\text{LEFT_OF}(x_1, x_2)]$ and $[\lambda(x_1, x_2)\text{ABOVE}(x_1, x_2)]$ would not be isomorphic under the given A-MAP.

3.3.5 Literally Similar and Analogically Similar Structural Descriptions

Having given the definitions of description comparison types above, we now give characterizations for when two structural descriptions are found literally similar and analogically similar. An equivalent-statement description comparison captures aspects literally true of corresponding objects, whether properties or relations. Therefore, such a description comparison identifies literal similarity across two descriptions. When such a description comparison puts into correspondence *all* aspects of each description (i.e., in a total-cover description comparison), we say that the two descriptions are “literally identical,” and when the comparison is *not* a total-cover description comparison or if the comparison is on two subdescriptions of larger descriptions, then we say that the (larger) descriptions are “literally similar.” An object-abstracted description comparison, on the other hand, captures what is literally true or isomorphic between the descriptions such that the only aspects considered are relations, and not properties. In accordance with our earlier discussion of the continuum of comparison types between literal similarity and analogy (as proposed in [Gentner 1983]), we see that in an object-abstracted description comparison, the more *equivalent* corresponding descriptive statements identified by the comparison, the more the comparison is taken as one of literal similarity, and the more *isomorphic* corresponding descriptive statements, the more the comparison is taken as one of analogical similarity. (Such sets of equivalent and isomorphic statements can be identified by equivalent-statement and isomorphic-statement description comparisons, respectively, on *subdescriptions* of each.) As with the definition of an equivalent-statement description comparison, when such a description comparison puts into correspondence all relations of each description, we say that the two descriptions are “analogues” of each other, and when all relations of each description are not put into correspondence, or if the comparison is on two subdescriptions of larger descriptions, then we say that the (larger) descriptions are “analogically similar.”

3.4 APPLICATION AND CHARACTERIZATION OF THE ABSTRACTIONAL CONCEPT MAPPING THEORY

We now discuss some computational aspects of the Abstractional Concept Mapping Theory, and present some further, undeveloped aspects characterizing our model.

3.4.1 Searching For Comparison Theories

Given that concept spaces are infinite (see Figure 9), the number of possible A-MAPS constructed between the domain vocabularies of a pair of structural descriptions is infinite. Therefore, heuristic methods are needed for the development of an effective procedure that generates A-MAPS identifying literal or analogi-

cal similarity between a pair of descriptions within a reasonable amount of compute time. We have made a first attempt of such an algorithm, heuristically-based on two aspects of the component descriptions: (a) the use of descriptive statements for heuristically determining object correspondences, and (b) the length of lambda expressions representing concepts for heuristically determining which concepts to consider first in the mapping (see [Dierbach 1991]).

3.4.2 Adequacy of Base-Concept A-MAPS

We identify a special set of concepts of a given concept space as those concepts represented by lambda expressions whose body is an atomic formula, e.g., $[\lambda(x_1, x_2)\text{LEFT_OF}(x_1, x_2)]$. We refer to such a concept as a “base concept.” In order to reduce the compute time in the generation of candidate A-MAPS between a given pair of structural descriptions, we are currently considering the adequacy of A-MAPS putting into correspondence only base concepts of one concept space with arbitrary concepts of another (referred to as a “base-concept A-MAP”) in identifying literal or analogical similarity between two descriptions. Such an A-MAP results in a correspondence of the predicate constants of each domain vocabulary with arbitrary concepts of the other, and thus results in a greatly simplified search method as compared to the correspondence of arbitrary concepts to arbitrary concepts.

3.4.3 Recursive Application of Description Comparisons

One of the distinctive features in the way that description comparisons have been defined is that comparison-based descriptions, by definition, are also structural descriptions. Thus, comparison-based descriptions are structural descriptions in their own right (although generally represent descriptions abstracted from a pair of descriptions grounded in the background knowledge of \mathcal{M}). This allows comparisons to be made between comparison-based descriptions, and thus gives a means of comparing two proposed comparison theories for a given pair of structural descriptions. The immediately evident application of such comparisons would be to determine, for two proposed comparison theories over the same component descriptions, if either: (a) the comparison theories are isomorphic, and (b) if one comparison theory subsumes the other. We only mention such possibilities here, and leave the details of such an application of description comparisons for future investigation.

3.4.4 \mathcal{L} -Substituted Comparison-Based Descriptions and \mathcal{M} -Axiomatizations

Initially, the comparison-based description of a given description comparison is given as an A -based structural description, i.e., such that the vocabulary of the initial description contains only indefinite constants, and no definite constants. It may be, however, that a given indefinite predicate constant P_1 maps to identical definite concepts, such as the definite concept

$[\lambda(x_1, x_2)\text{LEFT_OF}(x_1, x_2)]$. In this case, predicate constant P_1 can be naturally replaced by the predicate LEFT_OF . We refer to comparison-based descriptions in which such indefinite constants have been replaced by definite constants in the initial description as an “ \mathcal{L} -substituted comparison-based description.” Given that the comparison-based description may contain G-formulas found common to each description that are meaning formulas (e.g., $\forall x\forall y[[\text{DOMAIN_OBJ}(x) \wedge \text{DOMAIN_OBJ}(y) \wedge P_1(x, y)] \Rightarrow \sim P_1(y, x)]$), such meaning formulas, although necessarily part of the comparison theory when P_1 remains an indefinite constant (given that no meaning formulas exist in \mathcal{M} for indefinite constants), are unnecessary when P_1 is replaced by the definite constant LEFT_OF , given that such meaning formulas exist in \mathcal{M} and can be left as implied. Therefore, whenever a given comparison theory contains definite constants, an axiomatization of the comparison theory can be given which is potentially smaller than the original initial description of the comparison-based description (also seen as an axiomatization of the comparison theory). We refer to such an axiomatization as an \mathcal{M} -axiomatization.

The second usefulness of \mathcal{L} -substituted comparison-based descriptions is that such descriptions provide a means of *explanation* of the comparison, given that statements within the associated comparison theory contain terminology (i.e., vocabulary) of the background knowledge.

4 CONCLUSION

4.1 FUTURE WORK

There are many issues to be explored within the work presented here. We summarize some of these below:

- The development of a method of sound analogical inference based on the method of analogical matching provided by our model.
- The development of heuristic methods for the effective generation of A-MAPS.
- A formal characterization of the adequacy of base-concept A-MAPS.
- Investigation of how natural language forms of explanation of description comparisons can be generated from \mathcal{M} -axiomatized comparison theories.
- Investigation of the incorporation of the Abstractional Concept Mapping Theory into models of natural language understanding for the interpretation of metaphors.

4.2 SUMMARY

We have presented here a general, domain-independent computational model of analogical matching that allows for the representation of arbitrary comparisons between any two given first-order theories of logic by an abstracted first-order theory (and associated mappings). The model provides for a true semantic mapping in that such comparisons can be constructed regardless of the vocabulary used within each description or the particular syntactic form of individual statements. We see such a model of analogical matching as necessary before a sufficiently general model of analogical inference can be developed.

A unique aspect of our approach has been the pragmatic separation of background knowledge and "situation-specific" knowledge, enabling a clear distinction between corresponding equivalent statements, identifying identical aspects between descriptions, and corresponding isomorphic statements, identifying analogous aspects between the two. However, it is the explicit representation and mapping of *concepts* (as represented by lambda expressions) we see as the most unique contribution of our approach, providing a basis for both a theory of analogy and a theory of metaphor.

Acknowledgements

We wish to thank Marcelo Jenkins for insightful reading and fruitful discussion of this work.

References

- [Araya 1983] Araya, Agustin. A. "Learning By Controlled Transference of Knowledge Between Domains." In: *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, August, 1983 (IJCAI-83), pp. 439-443.
- [Burstein 1983] Burstein, M. H. "A Model of Learning by Incremental Analogical Reasoning and Debugging." In: *Proceedings of the National Conference on Artificial Intelligence*, August, 1983 (AAAI-83), pp. 45-48. American Association for Artificial Intelligence, Menlo Park, CA, 1983.
- [Carbonell 1983] Carbonell, J. G. "Derivational Analogy and its Role in Problem Solving." In: *Proceedings of the National Conference on Artificial Intelligence*, August, 1983 (AAAI-83), pp. 64-69. American Association for Artificial Intelligence, Menlo Park, CA, 1983.
- [Carbonell and Minton 1985] Carbonell, J. G. and Minton, Steven. "Metaphor and Commonsense Reasoning." In: *Formal Theories of the Commonsense World*, pp. 405-426. J. Hobbs and R. Moore (eds.) Ablex Publishing Company, Norwood, NJ, 1985.
- [Davies and Russell 1987] Davies, Todd R. and Russell, Stuart J. "A Logical Approach to Reasoning by Analogy." In: *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, August, 1987 (IJCAI-87), pp. 264-270.
- [Dierbach 1991] Dierbach, Charles. "Abstractional Concept Mapping: A Formal Basis for Analogical Reasoning." Technical Report 91-05 (Ph.D. thesis). Department of Computer and Information Sciences, University of Delaware, Newark, DE, 19716.
- [Douglas and Moran 1983] Douglas, Sarah A. and Moran, Thomas P. "Learning Operator Semantics by Analogy." In: *Proceedings of the National Conference on Artificial Intelligence*, August, 1983 (AAAI-83), pp. 100-103. American Association for Artificial Intelligence, Menlo Park, CA, 1983.
- [Falkenhainer 1987] Falkenhainer, Brian. "An Examination of the Third Stage in the Analogy Process: Verification-Based Analogical Learning." In: *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, August, 1987 (IJCAI-87), pp. 260-263.
- [Falkenhainer, Forbus and Gentner 1989] Falkenhainer, Brian, Forbus, Kenneth D., and Gentner, Dedre. "The Structure-Mapping Engine: Algorithms and Examples." *Artificial Intelligence* 41(1):1-63 (Nov. 1989).
- [Gentner 1983] Gentner, Dedre. "Structure-Mapping: A Theoretical Framework for Analogy." *Cognitive Science*, 7(2):155-170 (April-June 1983).
- [Gentner 1988] Gentner, Dedre. "Analogical Inference and Analogical Access." In: *Analogica (Proceedings of the First Workshop on Analogical Reasoning)* (Dec. 1985), pp. 63-88. Armand Prieditidis (ed.). Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1988.
- [Gick and Holyoak 1983] Gick, Mary L. and Holyoak, Keith J. "Schema Induction and Analogical Transfer." *Cognitive Psychology*, 15(1):1-38 (Jan. 1983).
- [Greiner 1988] Greiner, Russell. "Learning by Understanding Analogies." *Artificial Intelligence*, 35(1):81-125 (May 1988).
- [Indurkhy 1985] Indurkhy, Bipin. *A Computational Theory of Metaphor Comprehension and Analogical Reasoning*. Technical Report 85-001 (Ph.D. thesis). Computer Science Dept., Boston University, Boston, MA, 1985.
- [Indurkhy 1986] Indurkhy, Bipin. *Metaphors and Analogies: An Interdisciplinary Perspective*. Technical Report 86-012 (Dec. 1986). Computer Science Dept., Boston University, Boston, MA, 1986.
- [Indurkhy 1986a] Indurkhy, Bipin. "Constrained Semantic Transference: A Formal Theory of Metaphors." *Synthese* 68(3):515-551 (Sept. 1986).
- [Indurkhy 1988] Indurkhy, Bipin. "Constrained Semantic Transference: A Formal Theory of Metaphors." In: *Analogica (Proceedings of the First Workshop on Analogical Reasoning)* (Dec. 1985), pp. 129-157. Armand Prieditidis (ed.). Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1988.
- [Kedar-Cabelli 1988] Kedar-Cabelli, Smadar. "Toward A Computational Model of Purpose-Directed Analogy." In: *Analogica (Proceedings of the First Workshop on Analogical Reasoning)* (Dec. 1985), pp. 89-107. Armand Prieditidis (ed.). Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1988.
- [Shinn 1988] Shinn, Hong S. "Abstractional Analogy: A Model of Analogical Reasoning." In: *Proceedings Case-Based Reasoning Workshop*, May 1988, pp. 370-387. Morgan Kaufmann Publishers, Inc., Palo Alto, CA, 1988.
- [Tourangeau and Sternberg 1982] Tourangeau, Roger and Sternberg, Robert J. "Understanding and Appreciating Metaphor." *Cognition*, 11(3):203-244 (May 1982).
- [Winston 1980] Winston, Patrick H. "Learning and Reasoning by Analogy." *Communications of the Association for Computing Machinery (CACM)*, 23(12):689-703 (Dec. 1980).

The Complexity of Concept Languages

Francesco M. Donini
Maurizio Lenzerini, Daniele Nardi
Dipartimento di Informatica e Sistemistica
Università di Roma “La Sapienza”
via Salaria 113, I-00198, Roma, Italy

Werner Nutt
German Research Center for
Artificial Intelligence (DFKI)
PO 2080, D-6750 Kaiserslautern, Germany
Email: nutt@dfki.uni-kl.de

Abstract

Concept languages provide a means for expressing knowledge about hierarchies of concepts, i.e. classes of objects with common properties. The basic reasoning tasks to be performed on concepts are satisfiability checking and subsumption checking. Although concept languages have been widely used in AI-systems, only in rare cases complete algorithms have been known. The few complexity results that have been obtained gave mostly lower bounds and were not related to each other. In this paper we consider a family of languages, called \mathcal{AL} -languages, which covers most of the concept languages considered in the literature. The paper has two functions. First, we develop a general technique for checking satisfiability and subsumption of \mathcal{AL} -concepts, which is based on the tableau calculus for predicate logic. Second, we use this algorithmic technique to give precise upper and lower bounds for the \mathcal{AL} -languages, thus providing a complete analysis of the computational complexity of the satisfiability and the subsumption problem for concept languages.

1 INTRODUCTION

Concept languages have been investigated mainly in the field of knowledge representation, following the ideas initially embedded in many frame-based and semantic-network-based languages, especially the KL-ONE language [BS85]. However, several formalisms of this kind are now being considered within the realm of data bases and logic programming, with the aim of enriching the expressivity of existing data models and logic programming languages with object-oriented features (see [Be88, AN86]).

Concept languages can be given a Tarski style declarative semantics that allows them to be conceived as

sublanguages of predicate logic [BL84]. A concept is built up of two kinds of symbols, primitive concepts and primitive roles. An interpretation interprets them as subsets of a domain and binary relations over the domain. These primitives can be combined by various language constructs (such as intersection, union, role quantification, etc.) yielding complex concepts, which again are interpreted as subsets of the domain. Different languages are distinguished by the different sets of constructs they provide.

To give examples we suppose that **person** and **female** are primitive concepts, and **child** and **female_relative** are primitive roles. Using the set theoretical connectives intersection and complement, we can describe the class of “persons that are not female” by the concept

$$\text{person} \sqcap \neg \text{female}.$$

Most languages provide quantification over roles that allows for instance to describe the classes of “individuals whose children are all female” and “individuals having a female child” by the concepts

$$\forall \text{child}. \text{female} \quad \text{and} \quad \exists \text{child}. \text{female}.$$

Number restrictions on roles denote sets of individuals having at least or at most a certain number of fillers for a role. For instance,

$$(\geq 3 \text{female_relative}) \sqcap (\leq 2 \text{child})$$

denotes the class of “all individuals having at least three friends and at most two children.” Intersection can also be used as a role forming construct. For instance, the intersection

$$\text{child} \sqcap \text{female_relative},$$

intuitively yields the role “daughter.”

The basic reasoning tasks on concepts are satisfiability and subsumption checking. A concept is unsatisfiable if it denotes the empty set in every interpretation, and is satisfiable otherwise. A concept C is subsumed by a concept D if in every interpretation C denotes a subset of the set denoted by D . For a long time, the KL-ONE community was content with sound, but incomplete

subsumption algorithms. Such an algorithm delivers a correct answer when given C and D such that C is not subsumed by D , but sometimes fails to recognize that one concept is subsumed by another one.

Complexity analysis of the subsumption problem originated with the paper [BL84] by Brachman and Levesque, which provides a polynomial algorithm for a very limited language, called \mathcal{FL}^- , and shows that for the seemingly slightly more expressive language \mathcal{FL} subsumption is co-NP-hard. Nebel [Ne88] identified other constructs that give rise to co-NP-hard subsumption problems. Neither [BL84] nor [Ne88] give algorithms for the co-NP-hard languages. The first nontrivial subsumption algorithm was devised by Schmidt-Schauß and Smolka [SS91] for the language \mathcal{ALC} , an extension of \mathcal{FL} . They proved that unsatisfiability and subsumption in \mathcal{ALC} are PSPACE-complete and identified a sublanguage with co-NP-complete unsatisfiability problem.

In the present paper, we consider a family of languages, called \mathcal{AL} -languages, which includes most of the concept languages considered in the literature. In the simplest \mathcal{AL} -language, called \mathcal{AL} , *concepts* (denoted by the letters C and D) are built out of *primitive concepts* (denoted by the letter A) and *primitive roles* according to the syntax rule

$$C, D \longrightarrow A \mid T \mid \perp \mid C \sqcap D \mid \neg A \mid \forall R.C \mid \exists R.T$$

where R denotes a *role*, that in \mathcal{AL} is always primitive (more general languages provide a constructor for role intersection).

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of \mathcal{I}) and a function $.^{\mathcal{I}}$ (the *interpretation function* of \mathcal{I}) that maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that

$$\begin{aligned} T^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\ \perp^{\mathcal{I}} &= \emptyset \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\ (\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b. (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\ (\exists R.T)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b. (a, b) \in R^{\mathcal{I}}\}. \end{aligned}$$

Obviously, an interpretation function is already determined by the way it interprets primitive concepts and roles. An interpretation \mathcal{I} is a *model* for a concept C if $C^{\mathcal{I}}$ is nonempty. A concept is *satisfiable* if it has a model and *unsatisfiable* otherwise. We say C is *subsumed* by D if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every interpretation \mathcal{I} , and C is *equivalent* to D if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every interpretation \mathcal{I} .

More general languages are obtained by adding to \mathcal{AL} the following constructs:

- *union* of concepts (indicated by the letter \mathcal{U}), written as $C \sqcup D$, and defined by

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}};$$

- full *existential quantification* (indicated by the letter \mathcal{E}), written as $\exists R.C$, and defined by

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b. (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\};$$

- *complement* of non-primitive concepts (indicated by the letter \mathcal{C}), written as $\neg C$, and defined by

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}};$$

- *number restrictions* (indicated by the letter \mathcal{N}), written as $(\geq n R)$ and $(\leq n R)$, where n ranges over the nonnegative integers coded in unary (i.e., the integer n is represented by a string of length n), and defined by

$$(\geq n R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \geq n\},$$

and

$$(\leq n R)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \leq n\},$$

respectively;

- intersection of *roles* (indicated by the letter \mathcal{R}), written as $Q \sqcap R$, where Q and R are arbitrary roles, and defined by

$$(Q \sqcap R)^{\mathcal{I}} = Q^{\mathcal{I}} \cap R^{\mathcal{I}}.$$

We consider all combinations of the above constructs in concept languages. Every \mathcal{AL} -language is named by a string of the form

$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{R}],$$

where a letter in the name stands for the presence of the corresponding construct in the language. Observe that the combination of union and full existential quantification gives the possibility to express complements of concepts, and conversely, union and full existential quantification can be expressed using complements. Hence, without loss of generality we will assume that union and full existential quantification are available in every language that contains complements, and vice versa. In language names we will use the letter \mathcal{C} instead of $\mathcal{U}\mathcal{E}$. It follows that there are 16 pairwise non-equivalent \mathcal{AL} -languages, which form a lattice, whose bottom element is \mathcal{AL} and whose top element is \mathcal{ALCNR} .

The present paper features two main results. First, we define a general technique for checking unsatisfiability (and therefore subsumption, since C is subsumed by D if and only if $C \sqcap \neg D$ is unsatisfiable) in \mathcal{AL} -languages, thus providing complete algorithms for the basic inferences in concept languages. Following an idea presented in [SS91], our technique relies on a set of rules, which closely resemble the rules of the tableau calculus for first order logic. In fact, if one translates concepts into predicate logic formulas, and applies to them the tableaux calculus with a suitable control strategy, one obtains essentially the calculus described here.

Second, we give a detailed complexity analysis of both unsatisfiability and subsumption for \mathcal{AL} -languages. We have classified 15 of the 16 languages with respect to the complexity of both problems. The only exception is \mathcal{ALEN} , where we can say that unsatisfiability and subsumption are in PSPACE, and both are co-NP-hard (this follows from the results reported in [Ne88]) and NP-hard. Notice that for only one of these languages, namely \mathcal{ALC} , both the upper and the lower complexity bound for the two problems were previously known. Since for all \mathcal{AL} -languages but one the upper and lower bounds we give coincide, one can say that our algorithms are optimal for the problems they solve.

The rest of the paper is organized as follows. In Section 2, we describe the calculus for deciding the unsatisfiability of concepts. In Sections 3 to 6 we classify the family of \mathcal{AL} -languages with respect to the unsatisfiability and the subsumption problem. We consider in turn PSPACE-complete, NP-complete, co-NP-complete, and polynomial languages. Complete proofs of the results are given in [DL*91].

2 A CALCULUS FOR \mathcal{AL} -LANGUAGES

In this section we provide the basic elements of a rule based calculus for checking the satisfiability of concepts in \mathcal{AL} -languages. The calculus operates on constraints consisting of variables, concepts and primitive roles. To keep the number of rules small, we assume concepts to be in a normal form, defined as follows: a concept is called *simple* if it contains only complements of the form $\neg A$, where A is a primitive concept. Simple concepts are the analogue of logical formulae in negation normal form. Arbitrary concepts can be rewritten into equivalent simple concepts in linear time, using De Morgan's laws, the duality between universal and existential quantification, and the fact that a negated "atleast" restriction can be expressed as an "atmost" restriction, and vice versa.

We assume that there exists an alphabet of variable symbols, which will be denoted by the letters x , y , and z . A constraint is a syntactic object of one of the forms

$$x:C, \quad xPy,$$

where C is a simple concept and P is a primitive role. Intuitively, $x:C$ represents the constraint that x is in the interpretation of C , while xPy represents the constraint that the pair (x, y) is in the interpretation of P .

We extend the semantics of concepts to constraints. Let \mathcal{I} be an interpretation. An \mathcal{I} -assignment is a function α that maps every variable to an element of $\Delta^{\mathcal{I}}$. We say that α satisfies $x:C$ if $\alpha(x) \in C^{\mathcal{I}}$, and α satisfies xPy if $(\alpha(x), \alpha(y)) \in P^{\mathcal{I}}$. A constraint c is *satisfiable* if there is an interpretation \mathcal{I} and an \mathcal{I} -assignment

α such that α satisfies c . A *constraint system* S is a finite, nonempty set of constraints. An \mathcal{I} -assignment α satisfies a constraint system S if α satisfies every constraint in S . A constraint system S is *satisfiable* if there is an interpretation \mathcal{I} and an \mathcal{I} -assignment α such that α satisfies S .

Proposition 2.1 *A concept C is satisfiable if and only if the constraint system $\{x:C\}$ is satisfiable.*

Our calculus starts with a constraint system $S = \{x:C\}$ where C is a simple concept. Subsequent steps add constraints to S until either a contradiction is generated or an interpretation satisfying C can be obtained from the resulting system.

Let us introduce some notation to help us specifying the rules of the calculus. We say that role Q is a *subrole* of R and write $Q \preceq_R R$, if $Q^{\mathcal{I}} \subseteq R^{\mathcal{I}}$ for every interpretation \mathcal{I} . If $Q = P_1 \sqcap \dots \sqcap P_k$ and $R = P'_1 \sqcap \dots \sqcap P'_l$, then Q is a subrole of R if and only if for every primitive role P'_j occurring in R there is a role P_i occurring in Q such that $P_i = P'_j$.

Let S be a constraint system and $R = P_1 \sqcap \dots \sqcap P_k$ be a role. We say that xRy holds in S if the constraints xP_1y, \dots, xP_ky are in S . We denote by $n_{R,S}(x)$ the number of variables y such that xRy holds in S , i.e. $n_{R,S}(x) = |\{y \mid xRy \text{ holds in } S\}|$. We denote by $[y/z]S$ the constraint system obtained from S by replacing each occurrence of y by z . Finally, we say that the replacement of y by z is *safe* in S if for every variable x and for every role R such that $x:(\geq n R)$ is in S and xRy, xRz hold in S , we have $n_{R,S}(x) > n$. Intuitively, the replacement of y by z is safe in S , if every number restriction that is satisfied in S remains satisfied in $[y/z]S$.

The calculus is defined by the following *completion rules*:

1. $S \rightarrow_{\sqcap} \{x:C_1, x:C_2\} \cup S$
if $x:C_1 \sqcap C_2$ is in S , and $x:C_1$ and $x:C_2$ are not both in S
2. $S \rightarrow_{\sqcup} \{x:D\} \cup S$
if $x:C_1 \sqcup C_2$ is in S , neither $x:C_1$ nor $x:C_2$ is in S , and $D = C_1$ or $D = C_2$
3. $S \rightarrow_{\exists} \{xP_1y, \dots, xP_ky, y:C\} \cup S$
if $x:\exists R.C$ is in S , $R = P_1 \sqcap \dots \sqcap P_k$, there is no z such that xRz holds in S and $z:C$ is in S , and y is a new variable
4. $S \rightarrow_{\forall} \{y:C\} \cup S$
if $x:\forall R.C$ is in S , xRy holds in S , and $y:C$ is not in S
5. $S \rightarrow_{\geq} \{xP_1y, \dots, xP_ky\} \cup S$
if $x:(\geq n R)$ is in S , $R = P_1 \sqcap \dots \sqcap P_k$, $n_{R,S}(x) < n$, and y is a new variable
6. $S \rightarrow_{\leq} [y/z]S$

if $x:(\leq n R)$ is in S , xRy and xRz hold in S , $n_{R,S}(x) > n$, and the replacement of y by z is safe in S .

Proposition 2.2 *If C is a simple concept, then there is no infinite chain of completion steps issuing from $\{x:C\}$.*

The termination of the completion rules is due to the fact that the concepts in constraints introduced by a rule application are always smaller than the concept in the constraint to which the rule has been applied. In addition the safeness condition in the \rightarrow_{\leq} -rule is needed to prevent the \rightarrow_{\geq} -rule, which introduces variables, and the \rightarrow_{\leq} -rule, which identifies variables, from looping.

We distinguish two kinds of completion rules, *deterministic* ones ($\rightarrow_{\forall}, \rightarrow_{\exists}, \rightarrow_{\forall}, \rightarrow_{\geq}$) and *nondeterministic* ones ($\rightarrow_{\exists}, \rightarrow_{\leq}$). The nondeterministic rules correspond to concept constructs that contain disjunction, when translated into logic. Obviously, the union of concepts is the analogue of the disjunction of formulae. Although less obvious, disjunction is also present in “atmost” restrictions.

Proposition 2.3 *Let S and S' be constraint systems. Then:*

1. *If S' is obtained from S by application of a deterministic rule, then S is satisfiable if and only if S' is satisfiable.*
2. *If S' is obtained from S by application of a nondeterministic rule, then S is satisfiable if S' is satisfiable. Furthermore, if a nondeterministic rule applies to S , then it can be applied in a way that it yields a constraint system S' such that S' is satisfiable if and only if S is satisfiable.*

A constraint system is *complete* if no completion rule applies to it. A *clash* is a constraint system having one of the following forms:

- $\{x: \perp\}$
- $\{x: A, x: \neg A\}$
- $\{x: (\leq 0 R), xP_1y, \dots, xP_ky\}$
where $R = P_1 \sqcap \dots \sqcap P_k$
- $\{x: (\geq m Q), x: (\leq n R)\}$
where $m > n$ and $Q \preceq_R R$.

A clash-free complete system S derived from $\{x:C\}$ represents a model of C , whose elements are the variables occurring in S . A variable y is in the extension of A if $y: A$ is in S , and a pair (x, y) is in the extension of P if xPy is in S . Conversely, a constraint system that contains a clash is clearly unsatisfiable.

Theorem 2.4 *Let C be a simple concept. Then C is satisfiable if and only if $\{x:C\}$ can be transformed with*

the completion rules into a complete constraint system that contains no clash.

On the basis of the above result, it is straightforward to turn the calculus into a decision procedure. To check a simple concept C for satisfiability, one has to generate all complete constraint systems derivable from $\{x:C\}$, which are, up to variable renaming, finitely many. If all these systems contain a clash, then C is unsatisfiable, otherwise it is satisfiable.

The completion calculus provides also a procedure to decide subsumption, since a concept C is subsumed by a concept D if $C \sqcap \neg D$ is unsatisfiable. For an \mathcal{AL} -language that allows for complements the subsumption problem is equivalent to the unsatisfiability problem because $C \sqcap \neg D$ is in the language if C and D are. In languages that do not allow for complements the subsumption problem is strictly more general than the unsatisfiability problem, since there always exists some concept D in the language such that there is no concept equivalent to $\neg D$ in the language.

In the next sections we will give upper complexity bounds for the subsumption problem in the various \mathcal{AL} -languages by devising special control structures for the completion calculus and by introducing specialized versions of the rules for strict sublanguages of \mathcal{ALCNR} . Notice that the set of rules defined above corresponds to the language \mathcal{ALCNR} , the top element of the lattice of \mathcal{AL} -languages. Upper complexity bounds for the subsumption problem apply to the unsatisfiability problem, too, since a concept is unsatisfiable if and only if it is subsumed by the empty concept \perp .

Analyzing the behaviour of the completion rules when applied to inputs from special \mathcal{AL} -languages we derive lower complexity bounds for the unsatisfiability problem in several \mathcal{AL} -languages, thus providing also lower bounds for the subsumption problem.

3 PSPACE-COMPLETE LANGUAGES

In this section we start with a control structure for the completion calculus that allows to check satisfiability of \mathcal{ALCNR} -concepts using quadratic space. Then we will turn to \mathcal{AL} -languages for which satisfiability is PSPACE-hard. It was shown in [SS91] that \mathcal{ALC} has this property. We will identify two other minimal PSPACE-hard \mathcal{AL} -languages, namely \mathcal{ALUR} and \mathcal{ALNR} .

3.1 A PSPACE-ALGORITHM FOR \mathcal{ALCNR}

As already pointed out in [SS91] complete constraint systems for \mathcal{ALC} may be of exponential size. For a polynomial space algorithm it is therefore crucial not

to keep an entire complete constraint system in the memory but to store only small portions of it at a time. To make this idea more precise we need the following definitions.

Let S be a constraint system and x, y, y' be variables occurring in S . We say that y is a *successor* of x if S contains a constraint of the form xPy . We say that y is a *neighbour* of y' if there is a variable x such that $y \neq y'$, and y and y' are successors of x .

Next we give transformation rules that build up portions of complete constraint systems. The *trace rules* consist of the \rightarrow_{\sqcap} -, \rightarrow_{\sqcup} -, \rightarrow_{\forall} - and the \rightarrow_{\leq} -rule together with the following two rules that replace the \rightarrow_{\exists} - and the \rightarrow_{\geq} -rule:

1. $S \rightarrow_{T_3} \{xP_1y, \dots, xP_ky, y:C\} \cup S$
if $x:\exists R.C$ is in S , $R = P_1 \sqcap \dots \sqcap P_k$,
there is no neighbour x' of x such that S contains a constraint of the form $x'P'y'$,
there is no z such that xRz holds in S and $z:C$ is in S ,
and y is a new variable
2. $S \rightarrow_{T_2} \{xP_1y, \dots, xP_ky\} \cup S$
if $x:(\geq n R)$ is in S ,
 $R = P_1 \sqcap \dots \sqcap P_k$, $n_{R,S}(x) < n$,
there is no neighbour x' of x such that S contains a constraint of the form $x'P'y'$,
and y is a new variable.

The trace rules are designed to produce for a variable x at most one successor that has itself successors. More restrictive trace rules have been defined in [SS91] for ACC . The ACC -trace rules introduce for every variable at most one successor at all. When dealing with the richer language ALCNR , however, such a control strategy would lead to an incomplete algorithm. In the present case, for every variable all successors must be present simultaneously at some stage of the computation, since only then the interplay of role intersections and number restrictions forces us to identify certain successors. This is important because, when identifying variables, the constraints imposed on them are combined, which may lead to clashes that otherwise would not have occurred.

Let C be a simple ALCNR -concept and T be a constraint system obtained from $\{x:C\}$ by application of the trace rules. We call T a *trace* of $\{x:C\}$ if no trace rule applies to T .

Proposition 3.1 Let C be a simple ALCNR -concept. Then:

1. The length of a trace rule derivation issuing from $\{x:C\}$ is bounded by the size of C .
2. Every complete constraint system extending $\{x:C\}$ can be obtained as the union of finitely many traces.

3. Suppose S is a complete constraint system extending $\{x:C\}$ and T is a finite set of traces such that $S = \bigcup_{T \in T} T$. Then S contains a clash if and only if some $T \in T$ contains a clash.

Part (1) of the above proposition would not be true if we had not assumed that numbers are coded as unary strings. If numbers were coded in binary then the number of successors of a given variable could be exponential in the size of the concept to be checked. Part (3) says that to detect clashes it suffices to inspect traces.

Based on Proposition 3.1 it is straightforward to write an algorithm that generates all complete constraint systems derivable from $\{x:C\}$ while keeping only one trace in memory at a time. If pointers are used to represent the subconcepts of C occurring in a trace the algorithm one needs space at most quadratic in the size of C to store a trace and the necessary control information. Examples of such algorithms have been given for sublanguages of ALCNR in [SS91, HN*90]. From these considerations we conclude the main result of this subsection.

Theorem 3.2 Satisfiability and and subsumption of ALCNR -concepts can be decided with quadratic space.

3.2 UNSATISFIABILITY IN ACUR

We will show that unsatisfiability in ACUR is PSPACE-hard by a reduction of the unsatisfiability problem for ACC . The reduction is based on the observation that subroles can be used to simulate full existential quantification. Since we will use this technique in later sections as well, we define it for the entire language ALCNR .

We transform every simple ALCNR -concept C into an ACUNR concept \tilde{C} such that \tilde{C} is satisfiable if and only if C is satisfiable. Recall that by definition of ACUNR the concept \tilde{C} contains no subconcept of the form $\exists R.D$ where $D \neq T$.

Let C be a simple ALCNR -concept. Suppose that for every subconcept D of C there is a primitive role P_D that does not occur in C . We define a function $\tilde{\cdot}$ that maps every subconcept D of C to a concept \tilde{D} . The function $\tilde{\cdot}$ is given by the equations

$$\begin{aligned}\widetilde{\exists R.D} &= \exists(R \sqcap P_D).T \sqcap \forall(R \sqcap P_D).\tilde{D} \\ \widetilde{\forall R.D} &= \forall R.\tilde{D} \\ \widetilde{D_1 \sqcap D_2} &= \tilde{D}_1 \sqcap \tilde{D}_2 \\ \widetilde{D_1 \sqcup D_2} &= \tilde{D}_1 \sqcup \tilde{D}_2\end{aligned}$$

and by $\tilde{D} = D$ if D is neither an intersection nor a union nor a concept of the form $\forall R.D'$ or $\exists R.D'$. We say that \tilde{C} is an \mathcal{E} -simulation of C .

Proposition 3.3 For every simple ALCNR -concept C one can compute in quadratic time an \mathcal{E} -simulation \tilde{C} .

Lemma 3.4 Let C be a simple ALCNR -concept and \tilde{C} be a simulation of C . Then C is satisfiable if and only if \tilde{C} is satisfiable.

Proof. Suppose that \tilde{C} is an \mathcal{E} -simulation obtained from C using primitive roles P_D .

If $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is an interpretation, then it is easy to see that $\tilde{D}^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every subconcept D of C . Therefore, C is satisfiable if \tilde{C} is satisfiable.

Conversely, we show that for every interpretation \mathcal{I} there exists an interpretation $\tilde{\mathcal{I}}$ over the same domain such that $\tilde{C}^{\tilde{\mathcal{I}}} = C^{\mathcal{I}}$. Let $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ be an interpretation. Then $\tilde{\mathcal{I}} = (\Delta, \cdot^{\tilde{\mathcal{I}}})$ is obtained from \mathcal{I} by defining $A^{\tilde{\mathcal{I}}} = A^{\mathcal{I}}$ if A is a primitive concept, $P^{\tilde{\mathcal{I}}} = \{(a, b) \in \Delta \times \Delta \mid b \in D^{\mathcal{I}}\}$ if $P = P_D$ for some subconcept D of C , and $P^{\tilde{\mathcal{I}}} = P^{\mathcal{I}}$ for the remaining primitive roles. Now it is straightforward to check that $\tilde{D}^{\tilde{\mathcal{I}}} = D^{\mathcal{I}}$ for every subconcept D of C . Thus \tilde{C} is satisfiable if C is satisfiable. \square

Corollary 3.5 Satisfiability of ALUR -concepts is PSPACE-hard.

Proof. Satisfiability of ALC -concepts can be reduced to satisfiability of ALUR -concepts, since for every simple ALC -concept C the \mathcal{E} -simulation \tilde{C} is in ALUR . \square

3.3 UNSATISFIABILITY IN ALNR

We will show that unsatisfiability in ALNR is PSPACE-hard. This sharpens a result by Nebel [Ne88], who showed that subsumption in ALNR is co-NP-hard. The proof is by a reduction of the satisfiability problem for quantified boolean formulae (QBF). We first reformulate QBF in an appropriate way (see also [SS91]). Then we reduce QBF to satisfiability in ALENR . Using \mathcal{E} -simulations we conclude from this result the claim.

3.3.1 Quantified Boolean Formulae

A *literal* is a nonzero integer. A *clause* is a nonempty finite set N of literals such that $l \in N$ implies $-l \notin N$. A *prefix* from m to n , where m and n are positive integers such that $m \leq n$, is a sequence

$$(Q_m m)(Q_{m+1} m + 1) \cdots (Q_n n),$$

where each Q_i is either “ \forall ” or “ \exists ”. A *quantified boolean formula* is a pair $P.M$, where, for some n , P is a prefix from 1 to n and M is a finite nonempty set of clauses containing only literals between $-n$ and n .

Let P be a prefix from m to n . A P -assignment is a mapping

$$\{m, m + 1, \dots, n\} \rightarrow \{\text{t}, \text{f}\}.$$

An assignment α satisfies a literal l if $\alpha(l) = \text{t}$ if l is positive and $\alpha(-l) = \text{f}$ if l is negative. An assignment α satisfies a clause if it satisfies at least one literal of the clause.

Let P be a prefix from m to n . A set \mathbf{A} of P -assignments is *canonical* for P if it satisfies the following conditions:

1. \mathbf{A} is nonempty
2. if $P = (\exists m)P'$, then all assignments of \mathbf{A} agree on m and, if P' is nonempty, $\{\alpha|_{\{m+1, \dots, n\}} \mid \alpha \in \mathbf{A}\}$ is canonical for P'
3. if $P = (\forall m)P'$, then
 - (a) \mathbf{A} contains an assignment that satisfies m and, if P' is nonempty, $\{\alpha|_{\{m+1, \dots, n\}} \mid \alpha \in \mathbf{A} \text{ and } \alpha(m) = \text{t}\}$ is canonical for P'
 - (b) \mathbf{A} contains an assignment that satisfies $-m$ and, if P' is nonempty, $\{\alpha|_{\{m+1, \dots, n\}} \mid \alpha \in \mathbf{A} \text{ and } \alpha(m) = \text{f}\}$ is canonical for P' .

A quantified boolean formula $P.M$ is *valid* if there exists a set \mathbf{A} of P -assignments that is canonical for P such that every assignment in \mathbf{A} satisfies every clause of M . It is well known that deciding the validity of quantified boolean formulae is a PSPACE-complete problem [GJ79].

3.3.2 The Reduction

Let $P.M$ be a quantified boolean formula where $P = (Q_1 1) \cdots (Q_m m)$ is a prefix from 1 to m and $M = \{M_1, \dots, M_n\}$. We translate $P.M$ into an ALENR concept

$$C_{P,M} = D_0 \sqcap C_0^1 \sqcap \dots \sqcap C_0^n.$$

Let A, B be primitive concepts, P_0, P_1, \dots, P_n be primitive roles, and $R = P_0 \sqcap P_1 \sqcap \dots \sqcap P_n$.

The concept D_0 is obtained from the prefix P using the equations

$$D_{l-1} = \begin{cases} (\exists R.A) \sqcap (\exists R.\neg A) \sqcap (\forall R.D_l) & \text{if } Q_l = \forall \\ \exists R.D_l & \text{if } Q_l = \exists \end{cases}$$

for $l \in 1..(m-1)$ and the equation

$$D_{m-1} = \begin{cases} (\exists R.A) \sqcap (\exists R.\neg A) & \text{if } Q_m = \forall \\ \exists R.T & \text{if } Q_m = \exists. \end{cases}$$

The concept C_0^i is obtained from the clause N_i as follows. Let

$$k = \max_{l \in N_i} |l|.$$

Then, for $l \in 1..(k - 1)$, define

$$C_{l-1}^i = \begin{cases} (\leq 2 P_i) \sqcap \exists P_i.(B \sqcap A) \sqcap \exists P_i.(\neg B \sqcap C_l^i) & \text{if } l \in N_i \\ (\leq 2 P_i) \sqcap \exists P_i.(B \sqcap \neg A) \sqcap \exists P_i.(\neg B \sqcap C_l^i) & \text{if } -l \in N_i \\ \forall P_i.C_l^i & \text{if neither } l \text{ nor } -l \text{ is in } N_i, \end{cases}$$

and define

$$C_{k-1}^i = \begin{cases} \forall P_i.A & \text{if } k \in N_i \\ \forall P_i.\neg A & \text{if } -k \in N_i. \end{cases}$$

We will give an intuitive explanation of the reduction. The concept $C_{P,M}$ is constructed in such a way that on the one hand canonical sets of assignments for P satisfying the clauses of M and on the other hand clash free complete constraint systems derived from $\{x:C_{P,M}\}$ can be translated into each other. Under this translation every assignment gives rise to a trace and vice versa. A variable y that occurs in a given trace with the constraint $y:A$ ($y:\neg A$) indicates that the corresponding assignment satisfies a certain positive (negative) literal.

To each number $l \in 1..m$ correspond the subconcepts of $C_{P,M}$ on level l , namely, D_{l-1} and $C_{l-1}^1, \dots, C_{l-1}^n$. The concept D_{l-1} encodes the l -th quantifier in the prefix P . Suppose S is a clash free complete constraint system obtained from $\{x:C_{P,M}\}$ and the constraint $y:D_{l-1}$ occurs in S . The concept D_{l-1} is defined in such a way that the completion rules will have created successors z, z' of y with the constraints $z:A, z':\neg A$ if the l -th quantifier is universal, and a single successor z without specifying whether it belongs to A or $\neg A$ if the l -th quantifier is existential. In both cases constraints will be added such that yRz (yRz and yRz') holds.

The concept C_0^i encodes the i -th clause M_i . We exploit the implicit disjunction present in the “atmost” restrictions to mimic logical disjunction in clauses. To see this, suppose clause N_i contains the positive literal l and the constraint $y:C_{l-1}^i$ occurs in the system S from above. The completion rules will have created successors v, v' of y together with the constraints

$$y:(\leq 2 P_i), yP_i v, v:B, v:A, \\ yP_i v', v':\neg B, v:C_l^i.$$

No subsequent application of rules will have identified v and v' , since otherwise S would contain a clash. In addition to v and v' a successor z (successors z, z') has been generated for the variable y such that yRz (yRz and yRz') holds. Since we have $R \preceq_R P_i$ and S contains $y:(\leq 2 P_i)$, either z and v or z and v' have been identified. The choice between v and v' corresponds to the logical “or” in the clause M_i . Identifying z and v corresponds to satisfying the literal l . When identifying z and v' we end up with constraints $z:C_l^i$ and

$z:D_l^i$, which corresponds to postponing the choice of a literal to be satisfied. A similar argument applies when two variables z, z' are present.

Along these ideas one can prove the correctness of our reduction.

Lemma 3.6 Suppose $P.M$ is a quantified boolean formula. Then $P.M$ is valid if and only if $C_{P,M}$ is satisfiable.

Corollary 3.7 Satisfiability of \mathcal{ALENR} -concepts is PSPACE-hard.

Proof. Since $C_{P,M}$ is in \mathcal{ALENR} for every quantified boolean formula $P.M$, Lemma 3.6 yields the claim. \square

Corollary 3.8 Satisfiability of \mathcal{ALNR} -concepts is PSPACE-hard.

Proof. For every simple \mathcal{ALENR} -concept C the \mathcal{E} -simulation \tilde{C} is in \mathcal{ALNR} . \square

3.4 SUMMARY

Since there exists a polynomial space algorithm for subsumption in \mathcal{ALCNR} , which is the top element of the lattice of \mathcal{AL} -languages, we know that subsumption and (un)satisfiability in all \mathcal{AL} -languages are in PSPACE. Moreover, (un)satisfiability is PSPACE-hard in \mathcal{ACC} , \mathcal{ALUR} , and \mathcal{ALNR} . It follows that for all extensions of these three languages subsumption and (un)satisfiability are PSPACE-complete.

Theorem 3.9 (PSPACE-Completeness)

Subsumption and (un)satisfiability are PSPACE complete problems for \mathcal{ACC} , \mathcal{ALUR} , \mathcal{ALNR} , \mathcal{ACCR} , \mathcal{ALCN} , \mathcal{ALENR} , \mathcal{ALUNR} , and \mathcal{ALCNR} .

4 NP-COMPLETE LANGUAGES

In this section we consider languages that do not contain constructs expressing logical disjunction—whether explicitly, like unions, or implicitly, like “atmost” restrictions. In the lattice of \mathcal{AL} -languages \mathcal{ALER} is the greatest element with this property.

Our first result shows that in \mathcal{ALER} subsumption checking is easier than in \mathcal{ACC} , \mathcal{ALUR} , \mathcal{ALNR} and their extensions. More precisely, whenever one \mathcal{ALER} -concept is subsumed by a second one, there exists in the completion calculus a proof for this fact which is of polynomial size. Since such a proof can be guessed in nondeterministic polynomial time, it follows that the problem of deciding subsumption between \mathcal{ALER} -concepts is in NP. Secondly, we prove that unsatisfiability in \mathcal{ALE} and \mathcal{ALR} is NP-hard. This result has a character different from former intractability results

for concept languages. The work in [BL84, Ne88, SS91] identified the disjunctive constructs role restriction, “atmost” restriction, and union that together with concept conjunction give rise to intractability. The NP-hardness of ALC and ALCR shows that the interplay of universal and existential quantifiers is a source of complexity, too. That both sources are of a different nature is illustrated by the fact that the first makes subsumption co-NP-hard and the latter makes it NP-hard.

Checking subsumption of ALER -concepts amounts to checking unsatisfiability of certain ALCR -concepts, having the form $C \sqcap \neg D$, where C, D are in ALER . Since ALCR does not allow for number restrictions we can refine the notion of trace. Recall that the trace rules for ALCNR , in order to cope with number restrictions, generate all successors of a selected variable, identify some of them, and then select a new variable among the successors to continue with. In contrast, the ALCR -trace rules will generate at most one successor for a variable.

The ALCR -trace rules consist of the \rightarrow_{\sqcap} , the \rightarrow_{\sqcup} , and the \rightarrow_{\forall} -rule given in Section 2, together with the rule

$$S \rightarrow_{\exists} \{xP_1y, \dots, xP_ky, y:C\} \cup S$$

if $x:\exists R.C$ is in S , there is no constraint of the form xPz in S , and y is a new variable.

The difference between the \rightarrow_{\exists} -rule and the \rightarrow_{\forall} -rule on the one hand and the \rightarrow_{\exists} -rule on the other hand is that the latter is applied only once for a variable x . We are thus compelled to make a nondeterministic choice amongst the constraints of the form $x:\exists R.C$. Throughout this section we implicitly refer to the ALCR -trace rules as *trace rules* and to ALCR -traces as *traces*.

4.1 AN NP-ALGORITHM FOR ALER

As already seen, checking subsumption between two ALER -concepts C and D is equivalent to checking the ALCR -concept $C \sqcap \neg D$ for unsatisfiability. We will show that, when C and D are ALER -concepts, due to the special form of the ALCR -concept $C \sqcap \neg D$, unions, that come in via complements of intersections, do not act as an additional source of complexity. For this purpose we set up the *s*-rule calculus that operates on sets of traces and behaves nondeterministically only for existential quantification. It is interesting to observe that the *s*-rules provide an alternative method for deciding the satisfiability of ALCR -concepts.

The *s*-rules are (T denotes a set of traces):

1. $\{T\} \cup T \xrightarrow{s} \{T'\} \cup T$
if $T \notin T$ and $T \rightarrow_* T'$ where $* \in \{\sqcap, \exists, \forall\}$
2. $\{T\} \cup T \xrightarrow{s} \{T', T''\} \cup T$

if $T \notin T$, $x:C \sqcup C'$ is in T , neither $x:C$ nor $x:C'$ is in T and $T' = \{x:C\} \cup T$, $T'' = \{x:C'\} \cup T$.

The *s*-rules eliminate the nondeterminism introduced by unions, since the two traces that can be obtained by an application of the \rightarrow_{\sqcup} -rule are put both into the new set of traces. The nondeterminism in choosing a constraint to which the \rightarrow_{\exists} -rule applies persists.

Proposition 4.1 *Let C be a simple ALCR -concept. Then every *s*-rule derivation starting with $\{\{x:C\}\}$ terminates.*

The *s*-rules provide a correct and complete calculus for checking the unsatisfiability of ALCR -concepts.

Theorem 4.2 *Let C be a simple ALCR -concept. Then C is unsatisfiable if and only if $\{\{x:C\}\}$ can be transformed by the *s*-rules into a set T such that each trace in T contains a clash.*

The above theorem suggests a method for deciding the unsatisfiability of an ALCR -concept C . The method generates a number of traces that is—in the general case—exponential in the size of C . This is not surprising, since unsatisfiability in ALCR is PSPACE-complete. When checking subsumption between ALER -concepts C and D , though, a better result can be achieved: we show that the number of traces generated by the application of *s*-rules is bounded by the size of D . Since traces are of polynomial size, the *s*-rules provide a nondeterministic polynomial time method for checking subsumption in ALER .

The proof relies heavily on the structure of traces that arise when checking subsumption. A constraint in a trace T is *closed* if it is of the form

- $x:C \sqcap D$, and both $x:C$ and $x:D$ are in T ;
- $x:\forall R.C$, and for all y such that xRy holds in T , $y:C$ is in T ;
- $x:\exists R.C$, and there exist y, P such that xPy is in T ;
- $x:C \sqcup D$, and either $x:C$ or $x:D$ is in T .

Intuitively, a constraint is closed if no trace rule applies to it. A constraint in T is *open* if it is not closed.

We say that a concept C contains *intersections* or *unions* if the symbols “ \sqcap ” or “ \sqcup ”, respectively, occur in the string C . Note that if D is an ALER -concept, then by rewriting $\neg D$ into a simple concept we obtain a concept D' that contains no intersection. Moreover, $\{x:C \sqcap D'\}$ is unsatisfiable if and only if $\{x:C, x:D'\}$ is unsatisfiable.

Lemma 4.3 *Let C, D be ALER -concepts and let D' be obtained by rewriting $\neg D$ to a simple ALCR -concept. Let T be a trace derived from $\{x:C, x:D'\}$.*

Then there is at most one open constraint $y:E$ in T such that E contains unions. If there is such a constraint then E contains no intersections.

From the preceding results we can conclude that subsumption between \mathcal{ALCR} -concepts is in NP.

Theorem 4.4 Subsumption in \mathcal{ALCR} can be decided in nondeterministic polynomial time.

Proof. Let C, D be \mathcal{ALCR} -concepts and let D' be obtained by rewriting $\neg D$ to a simple \mathcal{ALCR} -concept. By Theorem 4.2, C is subsumed by D if and only if $\{\{x:C, x:D'\}\}$ can be transformed with the s -rules into a set of traces each of which contains a clash. Since traces are of polynomial size, it suffices to show that any s -rule derivation starting with $\{\{x:C, x:D'\}\}$ leads to a set of traces T whose cardinality is bounded by the size of D' .

In fact, the only s -rule increasing the number of traces is the \rightarrow_U^S -rule. This rule can only be applied to an open constraint containing unions. By Lemma 4.3, its application diminishes the number of times that the union symbol occurs in open constraints at least by one. Therefore in any derivation, the \rightarrow_U^S -rule can be applied at most as many times as the union symbol occurs in D' . \square

4.2 NP-HARDNESS RESULTS

We first prove that unsatisfiability in \mathcal{ALC} is NP-hard. From this we conclude that unsatisfiability in \mathcal{ALR} is NP-hard, too.

Let C be an \mathcal{ALC} -concept. Since C does not contain any disjunctive construct, there is—up to variable renaming—only one complete constraint system that can be derived from $\{x:C\}$. It follows from Proposition 3.1 that C is unsatisfiable if and only if there exists a trace of $\{x:C\}$ that contains a clash. Therefore, to prove that an \mathcal{ALC} -concept is unsatisfiable, it is sufficient to guess one trace containing a clash out of the set of all possible traces.

However, the number of possible traces can be exponential. For example, it is easy to see that there are at least 2^n traces derivable from $\{x:C\}$, where C is of the form

$$\begin{aligned} & \exists P.A_1 \sqcap \exists P.B_1 \sqcap \\ & \forall P.(\exists P.A_2 \sqcap \exists P.B_2 \sqcap \\ & \quad \forall P.(\dots (\exists P.A_n \sqcap \exists P.B_n \sqcap \forall P.T) \dots)). \end{aligned}$$

To prove that deciding the unsatisfiability of \mathcal{ALC} -concepts is an NP-hard problem we reduce to it the *clause traversal problem*, which will be defined below. A *positive clause* is a finite set of positive integers. If $M = \{M_1, \dots, M_m\}$ is a finite set of positive clauses, then a *traversal* of M is a finite set of positive integers N such that $N \cap M_l$ is a singleton for $l \in 1..m$.

Proposition 4.5 Deciding for a finite set of positive clauses whether there exists a traversal is an NP-complete problem.

Proof. It is easy to see that the problem is a reformulation of the ONE-IN-THREE 3SAT problem (see [GJ79], p. 259). \square

The reduction consists in associating to every set of positive clauses M an \mathcal{ALC} -concept C_M such that M has a traversal if and only if C_M is unsatisfiable. The key idea is to relate the traversals of M to those traces of $\{x:C_M\}$ that contain a clash.

In the following we assume P to be a fixed primitive role. Let $M = \{M_1, \dots, M_m\}$ be a set of positive clauses, and let n be the maximum of the numbers occurring in the clauses of M . We translate M into the concept

$$C_M = C_1^1 \sqcap \dots \sqcap C_1^n \sqcap D_1.$$

The C_1^j 's are inductively defined through the equations

$$C_l^j = \begin{cases} \exists P.C_{l+1}^j & \text{if } j \in M_l \\ \forall P.C_{l+1}^j & \text{if } j \notin M_l \end{cases}$$

and

$$C_{m+l}^j = \begin{cases} \exists P.C_{m+l+1}^j & \text{if } j \in M_l \\ \forall P.C_{m+l+1}^j & \text{if } j \notin M_l \end{cases}$$

for $l \in 1..m$ and by the equation $C_{2m+1}^j = \top$. The concept D_1 is defined by the equations

$$D_l = \forall P.D_{l+1} \quad \text{and} \quad D_{m+l} = \forall P.D_{m+l+1}$$

for $l \in 1..m$ and by the equation $D_{2m+1} = \perp$.

Observe that to every number j occurring in M there corresponds a concept C_1^j and to every clause $M_l \in M$ there correspond the levels l and $m+l$ in the C_1^j 's. The number j is present in the l th clause if and only if there is an existential quantifier in the concept C_1^j at level l and at level $m+l$. As we shall see later the two layered construction of C_M is crucial for the correctness of the reduction. The concept D_1 is designed in such a way that in a trace a clash can only occur at level $2m+1$.

Next we identify those concepts in a trace of $\{x:C_M\}$ that will give us the numbers to construct a traversal of M . Let T be a trace and C be a concept. We say that C is *active in* T if C is of the form $\exists P.D$ and there are variables y, z such that T contains the constraints $y:C$, yPz , and $z:D$. Intuitively, an existentially quantified concept C is said to be active in T if the \rightarrow_{\exists} -rule has been applied to some constraint $y:C$ in T .

Lemma 4.6 Let T be a trace of $\{x:C_M\}$.

1. Suppose C_k^j is active in T . Then for all $l \in 1..k$ the concept C_l^j is active in T if it is of the form $\exists P.C_{l+1}^j$.

2. If T contains a clash, then for every $l \in 1..2m$ there exists exactly one j such that C_l^j is active in T .

Using these properties, we are able to prove the correctness of our reduction.

Theorem 4.7 A set \mathcal{M} of positive clauses has a traversal if and only if $C_{\mathcal{M}}$ is unsatisfiable.

Proof. Suppose $\mathcal{M} = \{M_1, \dots, M_m\}$ is a set of positive clauses and $C_{\mathcal{M}} = C_1^1 \sqcap \dots \sqcap C_1^n \sqcap D_1$ is its translation.

“ \Rightarrow ” Let N be a traversal of \mathcal{M} . To prove that $C_{\mathcal{M}}$ is unsatisfiable, we show that there exists a trace T of $\{x_1 : C_{\mathcal{M}}\}$ that contains a clash. To obtain T we inductively define a sequence of traces T_l of $\{x_1 : C_{\mathcal{M}}\}$, where $l \in 1..(2m+1)$, such that $T_l \subseteq T_{l+1}$ and $T_{2m+1} = T$. Let

$$T_1 = \{x_1 : C_1^j \mid j \in N\} \cup \{x_1 : D_1\}$$

and

$$\begin{aligned} T_{l+1} = & T_l \cup \{x_l P x_{l+1}\} \cup \{x_{l+1} : C_{l+1}^j \mid j \in N\} \\ & \cup \{x_{l+1} : D_{l+1}\}. \end{aligned}$$

Obviously, $T = T_{2m+1}$ contains a clash, because $D_{2m+1} = \perp$.

To show that T is a trace we prove that T_{l+1} can be obtained from T_l by application of the trace rules. Since N is a traversal, there is exactly one $k \in N$ such that $C_l^k = \exists P.C_{l+1}^k$. Hence,

$$T_l \rightarrow_{\exists} T'_l = T_l \cup \{x_l P x_{l+1}, x_{l+1} : C_{l+1}^k\}$$

by application of the \rightarrow_{\exists} -rule. Since we have $C_l^j = \forall P.C_{l+1}^j$ for $j \in N \setminus \{k\}$ and $D_l = \forall P.D_{l+1}$, it is easy to see that T_{l+1} can be obtained from T'_l by application of the \rightarrow_{\forall} -rule.

“ \Leftarrow ” If $C_{\mathcal{M}}$ is unsatisfiable, then there exists a trace T of $\{x : C_{\mathcal{M}}\}$ such that T contains a clash and for every $l \in 1..2m$ there exists exactly one j such that C_l^j is active in T .

Let

$$N = \{j \mid C_{m+l}^j \text{ is active in } T \text{ for some } l \in 1..m\}.$$

We show that N is a traversal. Let M_l be a clause in \mathcal{M} .

For every $l \in 1..m$, there exists a j such that C_{m+l}^j is active in T . Hence we have $j \in N$. Furthermore, $C_{m+l}^j = \exists P.C_{m+l+1}^j$ by the definition of active concepts. By construction of $C_{\mathcal{M}}$, we have that $j \in M_l$. Thus $j \in N \cap M_l$.

Suppose, there are i, j such that $i, j \in N \cap M_l$. Now we exploit the two-layered construction of $C_{\mathcal{M}}$. Since $i, j \in N$, by definition of N there are h, k such that

C_{m+h}^i and C_{m+k}^j are active in T . Since $i, j \in M_l$, by construction of $C_{\mathcal{M}}$ we have $C_l^i = \exists P.C_{l+1}^i$ and $C_l^j = \exists P.C_{l+1}^j$. By 4.6.1, we know that C_l^i and C_l^j are active in T . Hence $i = j$ by 4.6.2. \square

Corollary 4.8 Unsatisfiability in $\mathcal{A}\mathcal{E}$ is NP-hard.

Corollary 4.9 Unsatisfiability in $\mathcal{A}\mathcal{R}$ is NP-hard.

Proof. The claim follows by Lemma 3.4, since for every $\mathcal{A}\mathcal{E}$ -concept C the \mathcal{E} -simulation \tilde{C} is in $\mathcal{A}\mathcal{R}$. \square

4.3 SUMMARY

Combining the results in the preceding subsections we conclude the following theorem.

Theorem 4.10 (NP-Completeness) Subsumption and unsatisfiability are NP-complete problems for $\mathcal{A}\mathcal{E}$, $\mathcal{A}\mathcal{R}$, and $\mathcal{A}\mathcal{ER}$.

5 CO-NP-COMPLETE LANGUAGES

Since in \mathcal{ACU} intersection, union, and complement of primitive concepts are available, deciding the unsatisfiability of \mathcal{ACU} -concepts is at least as hard as deciding the unsatisfiability of a set of clauses in propositional logic, which is known to be a co-NP-complete problem. Conversely, it was shown in [SS91] that universal quantification and restricted existential quantification over roles do not increase the complexity of the problem, i.e., unsatisfiability in \mathcal{ACU} is co-NP-complete.

One might conjecture that subsumption in \mathcal{ACU} is harder, since it is equivalent to unsatisfiability of certain concepts containing both unions and full existential quantification. Now, it turns out that subsumption in an even larger language, namely \mathcal{ACUN} , has the same complexity as unsatisfiability in \mathcal{ACU} .

Since the language \mathcal{ACUN} is not closed under complements, subsumption checking in \mathcal{ACUN} amounts to checking the unsatisfiability of certain \mathcal{ACCN} -concepts. We will show that for \mathcal{ACCN} we can improve the general calculus given in Section 2 in such a way that “atleast”-constraints lead only to the introduction of a single new variable. Throughout this section we will assume that simple \mathcal{ACCN} -concepts do not contain subconcepts of the form $\exists P.T$. This is no loss of generality, because given an arbitrary \mathcal{ACCN} -concept we can replace every subconcept of the form $\exists P.T$ with the concept $(\geq 1 P)$, while preserving equivalence.

The quasi-completion rules consist of the \rightarrow_{\neg} -, \rightarrow_u -, \rightarrow_{\exists} -, \rightarrow_v -, and \rightarrow_{\leq} -rules together with the following revised \rightarrow_{\geq} -rule:

$S \rightarrow_{\geq} \{xPy\} \cup S$
 if $x:(\geq n P)$ is in S , $n \geq 1$,
 there is no z such that xPz is in S
 no other rules are applicable to S ,
 and y is a new variable.

Intuitively, this weak version of the \rightarrow_{\geq} -rule is sufficient because in the absence of role intersection for any variable the completion rules will impose the same constraints on all fillers of a given role. A constraint system is *quasi-complete* if no quasi-completion rule is applicable.

Lemma 5.1 *Let C be a simple $ALCN$ -concept. Then C is satisfiable if and only if $\{x:C\}$ can be transformed by the quasi-completion rules into a quasi-complete constraint system that contains no clash.*

Proof. A quasi-complete constraint system can be extended to a complete system by adding sufficiently many copies of variables to satisfy all “atleast” constraints. If the quasi-complete system does not contain a clash then the extension does not either. \square

Now we apply the quasi-completion rules to $ALCN$ -concepts that arise when checking subsumption in $ALUN$.

Lemma 5.2 *Let C, D be $ALUN$ -concepts and D' be obtained by rewriting $\neg D$ into a simple concept. Let S be a constraint system derived from $\{x:C \sqcap D'\}$ by means of the quasi-completion rules. Then the cardinality of S is almost quadratic in the size of $C \sqcap D'$.*

Intuitively, this lemma holds since there is no complex interaction between existential and universal quantifiers. If $\forall P.E$ is a subconcept of $C \sqcap D'$ then any existentially quantified subconcept of E is of the form $\exists P'.T$. Likewise, if $\exists P.E$ is a subconcept of $C \sqcap D'$ then any universally quantified subconcept of E is of the form $\forall P'.\perp$.

Corollary 5.3 *Nonsubsumption in $ALUN$ can be decided in nondeterministic polynomial time.*

Proof. Let C, D be $ALUN$ -concepts and D' be obtained by rewriting $\neg D$ into a simple concept. Then C is not subsumed by D if and only if $C \sqcap D'$ is satisfiable. By Lemma 5.1 this is equivalent to the fact that a clash free quasi-complete constraint system S can be derived from $\{x:C \sqcap D'\}$. By Lemma 5.2 the system S has cardinality quadratic in the size of C and D . Hence, S can be derived by a number of steps polynomial in the size of C and D . \square

Combining the fact that unsatisfiability in ALU is co-NP-hard and the fact that subsumption in $ALUN$ is in co-NP we obtain the following result.

Theorem 5.4 (Co-NP-Completeness)

Subsumption and unsatisfiability are co-NP-complete problems for ALU and $ALUN$.

6 POLYNOMIAL LANGUAGES

In Sections 4 and 5 we have shown that there exist two different sources of complexity of inferences in concept languages. On the one hand, full existential quantification—whether expressed explicitly, or implicitly through subroles—is responsible for constraint systems of exponential size. On the other hand, constructs that embody logical disjunction—like union or number restrictions in combination with subroles—make it necessary to generate an exponential number of constraint systems.

Consequently, one expects languages to be computationally tractable if they contain neither of the two sources of complexity. The largest such language is ALN . In this section we will sketch a polynomial size algorithm for ALN by showing that to decide subsumption between two ALN -concepts it suffices to check a linear number of constraint systems each of which is of linear size. We start with defining normal forms for ALN -concepts. We say that an ALN -concept is *admissible* if it is of the form

$$C_1 \sqcap \dots \sqcap C_n$$

such that none of the concepts C_i contains intersections. Every ALN -concept can be rewritten in polynomial time to an equivalent admissible concept using the rule

$$\forall P.(C \sqcap D) \longrightarrow (\forall P.C) \sqcap (\forall P.D).$$

Suppose that C, D are ALN -concepts, where D is admissible, that is $D = D_1 \sqcap \dots \sqcap D_n$ and none of the D_i contains intersections. To check whether C is subsumed by D it suffices to check whether $C \sqcap \neg D = C \sqcap \neg(D_1 \sqcap \dots \sqcap D_n)$ is unsatisfiable. Now, $C \sqcap \neg D$ is equivalent to the concept $(C \sqcap \neg D_1) \sqcup \dots \sqcup (C \sqcap \neg D_n)$. The latter is unsatisfiable if and only if each concept $C \sqcap \neg D_i$, $i \in 1..n$, is unsatisfiable. Let D'_i be the simple concept obtained by rewriting $\neg D_i$. Then D'_i is of the form

$$\exists P_1.\exists P_2.\dots.\exists P_m.E,$$

where E is of the form T , \perp , A , $\neg A$, $(\geq n P)$, or $(\leq n P)$. We call such a concept a *thread*.

It follows from the above considerations that to check subsumption of two ALN -concepts C, D it suffices to check for threads D'_1, \dots, D'_n , where n is bounded by the size of D , whether each $C \sqcap D'_i$ is unsatisfiable.

Proposition 6.1 *Let C be an ALN -concept and D' be a thread. Then:*

1. *There exists—up to variable renaming—only one quasi-complete constraint system that can be derived from $\{x:C \sqcap D'\}$.*

2. *The cardinality of every quasi-complete constraint system derivable from $\{x: C \sqcap D'\}$ is bounded linearly by the size of C and D' .*

Proof. Due to the particular form of $C \sqcap D'$, only deterministic quasi-completion rules will be applicable. This yields (1). In any constraint system derivable from $\{x: C \sqcap D'\}$ by means of the quasi-completion rules, a variable has at most one successor for every primitive role. This yields (2). \square

We have shown that to decide subsumption between \mathcal{ALN} -concepts it suffices to generate and check linearly many constraint systems each of which is of linear size. Taking into account the time for controlling the application of rules we obtain the following result on polynomial \mathcal{AL} -languages.

Theorem 6.2 (Polynomiality) *In \mathcal{AL} and \mathcal{ALN} , unsatisfiability can be decided in time $O(n^2)$, and subsumption can be decided in time $O(n^3)$.*

7 CONCLUSIONS

We have presented a calculus for checking satisfiability and subsumption in a wide class of concept languages covering most of the constructs proposed in the literature. Studying the computational properties of the calculus, we have obtained a complete picture of the complexity of concept languages. We have singled out two possible sources of complexity. First, complexity comes in via disjunction, which is reflected by co-NP-hardness of unsatisfiability and subsumption. Less obviously, full existential quantification is a source of complexity, too. This is reflected by NP-hardness results. For all \mathcal{AL} -languages where both sources of complexity are present, except \mathcal{ALEN} , we have shown that the unsatisfiability and the subsumption problem are PSPACE-hard.

Originally, complexity analysis of terminological reasoning was set up with the goal to identify languages for which subsumption can be decided in polynomial time [BL84]. Now it has turned out that practically all interesting constructs in concept languages lead to intractability.

One might conclude from these results that terminological reasoning in all its variants is infeasible. Such a conclusion would implicitly assume that the complexity analysis of subsumption is intended to restrict the practical use of concept languages to those where subsumption can be computed in polynomial time. However, it is our opinion that the study of the complexity of concept languages goes far beyond a mere classification of tractable and intractable languages.

First of all, the results developed so far refer to the computational complexity in the worst case, which represents only one aspect to be taken into account

when considering the practical use of concept languages. Notice that, as pointed out in [Ne90], another aspect that deserves further investigation is the characterization of the average cases occurring in practice. Second, the techniques used for the complexity analysis have provided the formal basis for the design of effective algorithms for computing subsumption and unsatisfiability in a large class of concept languages. Finally, in the design of deduction procedures for knowledge representation systems based on concept languages, one can take advantage of the knowledge about the complexity of subsumption, by isolating difficult cases and using specialized efficient algorithms whenever possible.

For all the above reasons, we believe that the research on the computational complexity of concept languages is a valuable support for the design of knowledge-based systems embedding forms of terminological reasoning.

Acknowledgements

This work was partly funded by the ESPRIT BRA 3012 (Compulog), the Italian CNR under Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, and the German BMFT under grant ITW 8903 0.

References

- [AN86] H. Aït-Kaci, R. Nasr. "LOGIN: a logic programming language with built-in inheritance." *Journal of Logic Programming*, 3, 1986.
- [Be88] C. Beeri. "Data models and languages for databases." *Proc. ICDT-88*, 1988.
- [BL84] R. J. Brachman, H. J. Levesque. "The tractability of subsumption in frame based description languages." *Proc. 4th AAAI*, Austin, Tex., 1984.
- [BS85] R. J. Brachman, J. Schmolze, "An overview of the KL-ONE knowledge representation system." *Cognitive Science*, 9 (2), 1985.
- [DL*91] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt. *The Complexity of Concept Languages*. DFKI Research-Report, Forthcoming.
- [GJ79] M. R. Garey, D. S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, Cal., 1979.
- [HN*90] B. Hollunder, W. Nutt, M. Schmidt-Schaub, "Subsumption algorithms for concept description languages." *Proc. 9th ECAI*, 1990.
- [Ne88] B. Nebel. "Computational complexity of terminological reasoning in BACK." *Artificial Intelligence*, 34(3), 1988.
- [Ne90] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, Springer LNAI 422, 1990.
- [SS91] M. Schmidt-Schaub, G. Smolka. "Attributive concept descriptions with complements." To appear in *Artificial Intelligence*, 47, 1991.

Rational Belief Revision* (Preliminary Report)

Jon Doyle
Laboratory for Computer Science
Massachusetts Institute of Technology
545 Technology Square
Cambridge, Massachusetts 02139, USA

Abstract

Theories of rational belief revision recently proposed by Alchourrón, Gärdenfors, Makinson, and Nebel illuminate many important issues but impose unnecessarily strong standards for correct revisions and make strong assumptions about what information is available to guide revisions. We reconstruct these theories according to an economic standard of rationality in which preferences are used to select among alternative possible revisions. By permitting multiple partial specifications of preferences in ways closely related to preference-based nonmonotonic logics, the reconstructed theory employs information closer to that available in practice and offers more flexible ways of selecting revisions. We formally compare this new conception of rational belief revision with the original theories, adapt results about universal default theories to prove that there is unlikely to be any universal method of rational belief revision, and examine formally how different limitations on rationality affect belief revision.

1 INTRODUCTION

One of the best developed formal theories of belief revision is the so-called AGM theory of Alchourrón, Gärdenfors, and Makinson [1985], which has been brought together with related results by the same authors in a book by Gärdenfors [1988]. (For convenience we will refer to results in the book rather than the original articles.) The AGM approach models belief states with sets of propositions and develops, motivates, and studies a small set of axioms that characterize how a rational agent should change its belief states when

* A preliminary version of the full paper was presented at the Third International Workshop on Nonmonotonic Reasoning, South Lake Tahoe, California, June 1990.

new beliefs are added, subtracted, or changed. The central result of this theory is that these axioms are equivalent to the existence of a complete preordering of all propositions according to their degree of *epistemic entrenchment* such that belief revisions always retain more entrenched propositions in preference to less entrenched ones.

The AGM theory of belief revision is not directly applicable to formalizing belief revision as practiced in artificial intelligence because it requires that revisions be deductively closed, infinite sets of propositions.¹ To better model AI practice, Nebel [1989] adapted the AGM theory so that finite sets of representing propositions mediate revisions. He then shows that the resulting revisions satisfy most of the AGM rationality axioms. In fact, Nebel shows how to define certain orders over propositions and sets of propositions, representing notions of *epistemic relevance*, so that selecting revisions to be maximal in these orders yields finite revisions which satisfy *all* the rationality axioms. He also proves the very satisfying result that revision by picking maximal consistent subsets can be just as rational as more elaborate forms of revision in which the reasoner retains only those beliefs following from all maximal consistent subsets. This makes mechanization of belief revision systems more practical since picking one consistent subset is much simpler than intersecting the consequences of all consistent subsets.

Unfortunately, even Nebel's theory does not completely succeed at formalizing practical notions of belief revision, even though it makes improvements in the AGM theory. Both theories suffer from unnecessarily strong standards for correct revisions, and from very strong assumptions about what information is available to guide revisions. Specifically, the sense of "rationality" postulated by these theories requires that rational revisions be unique (they do not permit equally acceptable alternative revisions). They also require total orderings of all propositions, even though most domains of knowledge formalized in AI systems are too

¹ Makinson [1987] discusses this and related problems with the "recovery" postulate of the AGM theory.

incomplete and ambiguous to supply such complete orderings.

We show that these two problems are related by reconstructing the theory of belief revision according to an economic standard of rationality. Where the AGM axioms refer only to logical properties of revised sets of beliefs and are motivated in terms of logical coherence and conservation of beliefs, we make the underlying motivation more explicit and precise by identifying various preferences guiding revisions and viewing rational revision as choosing the new belief state to be of maximal preferability according to these revision preferences. This change has two major effects. First, since there may be several alternatives of maximal preferability, our theory does not make the strong assumption of unique revisions. Second, the enlarged framework is closer to AI practice than its predecessors since it demands only partial information about revision preferences and permits these partial preferences to be combined and used in more flexible ways.

The plan of the paper is as follows. Sections 2 and 3 summarize in turn the formal theories and central results of Alchourrón, Gärdenfors, Makinson, and Nebel. Section 4 introduces the formal theory of rational revision in the economic sense and studies its impact on the AGM axioms. Determinism aside, rational revision turns out to be quite different from the AGM notion, which makes the generally plausible but occasionally dubious assumption that believing more is always better. Section 5 examines the informational requirements of rational revision and the AGM and Nebel theories. We argue that in practice the preferences available to guide belief revision consist of numerous partial and sometimes conflicting preferences, and show how the orderings assumed in the AGM and Nebel theories are too inflexible for many applications. We then present the formal theory of rational revision guided by multiple partial preferences, which is based on several principles for rationally aggregating partial preferences. This theory is formally similar to Doyle and Wellman's [1991] theory of rational default reasoning and to the economic theory of social choice [Arrow, 1963]. We prove that no method for belief revision based on partial preferences satisfies all the rationality conditions on preference aggregation. Finally, we examine how irrationalities in preference aggregation lead to violations of the AGM rationality axioms.

2 REVISING BELIEF STATES

The AGM formalization of belief revision may be summarized as follows, using an adaptation of the notations of [Alchourrón *et al.*, 1985] and [Nebel, 1989]. In the following we suppose that \mathcal{L} is a propositional language over the standard sentential connectives (\neg , \wedge , \vee , \rightarrow , \leftrightarrow), denote individual propositions by x , y , and z , and denote sets of propositions by A , B , and C .

We write \vdash to mean classical propositional derivability, and write Cn to mean the corresponding closure operator

$$Cn(A) \stackrel{\text{def}}{=} \{x \in \mathcal{L} \mid A \vdash x\}.$$

The AGM approach covers states of belief modeled in two ways: as deductively closed (but not necessarily consistent) sets of propositions, that is, propositional theories $A \subseteq \mathcal{L}$ such that $A = Cn(A)$, and also as belief bases that represent the beliefs contained in their deductive closure. Formally, we say that B is a base for A whenever $A = Cn(B)$. Naturally, a given theory can be represented by many different belief bases. The case of greatest practical interest is when the belief base B is finite (and small), but not every theory has a finite basis. We also use the same terminology if B is infinite, and even if $A = B$.

Each of these models of belief states gives rise to a different theory of belief revision. Most of the theoretical results, however, concern only the closed belief states. We will treat both approaches.

2.1 EXPANSIONS, CONTRACTIONS, AND REVISIONS

The AGM theory considers three types of operations on belief states. For each belief state A and proposition x we have:

Expansion: Expanding A with x , written $A + x$, means adding x to A and requiring that the result be a (possibly inconsistent) belief state.

Contraction: Contracting A with respect to x , written $A - x$, means removing x from A in such a way to result in a belief state.

Revision: Revising A with x , written $A \dot{+} x$, means adding x to A in such a way that the result is a consistent belief state.

Expansion is naturally defined in terms of the union of the set of beliefs and the new proposition. In the belief base model, we may take the expansion of A by x as this union itself. In this paper, however, we will focus on the case of closed belief states, and define the expansion to be the closure of this union

$$A + x \stackrel{\text{def}}{=} Cn(A \cup \{x\}).$$

Contraction and revision, on the other hand, have no single natural definitions, only the standard requirement that the change made be as small as possible so as to minimize unnecessary loss of knowledge.

Alchourrón, Gärdenfors, and Makinson [1985] formulate and motivate sets of rationality postulates that these operations should satisfy. The axioms for rational contractions are as follows. For each belief state A and propositions x and y :

- ($\dashv 1$) $A \dashv z$ is a belief state, and a theory whenever A is; (*closure*)
- ($\dashv 2$) $A \dashv z \subseteq A$; (*inclusion*)
- ($\dashv 3$) If $z \notin Cn(A)$, then $A \dashv z = A$; (*vacuity*)
- ($\dashv 4$) If $\vdash z$, then $z \notin Cn(A \dashv z)$; (*success*)
- ($\dashv 5$) If $\vdash z \leftrightarrow y$, then $A \dashv z = A \dashv y$; (*equivalence*)
- ($\dashv 6$) $A \subseteq Cn((A \dashv z) + z)$ whenever A is a theory; (*recovery*)
- ($\dashv 7$) $(A \dashv z) \cap (A \dashv y) \subseteq A \dashv (z \wedge y)$ whenever A is a theory;
- ($\dashv 8$) If $z \notin A \dashv (z \wedge y)$, then $A \dashv (z \wedge y) \subseteq A \dashv z$ whenever A is a theory.

The *closure* axiom ($\dashv 1$) says that contracting a theory yields a theory. The *inclusion*, *vacuity*, and *success* postulates ($\dashv 2$)-($\dashv 4$) state that removing a proposition does not introduce any new propositions, that no change occurs when one tries to remove a proposition that is not a consequence of the belief state, and that removed propositions are indeed removed, unless they are tautologies (in which case they are present in every theory, and so cannot be removed). The *equivalence* axiom ($\dashv 5$) (which is usually called the “preservation” axiom) states that the results of contraction do not depend on the syntactic form of the proposition removed; removing any logically equivalent proposition has the same effect. The *recovery* postulate ($\dashv 6$) states a conservation principle by requiring that contraction of a theory with respect to a proposition removes nothing that cannot be recovered by adding the proposition back in. Axioms ($\dashv 7$) and ($\dashv 8$) relate the contraction of a theory with respect to a conjunction to the contractions with respect to the individual conjuncts; they imply that retracting a conjunction preserves more information than retracting both of its conjuncts simultaneously.

Alchourrón, Gärdenfors, and Makinson also develop a parallel set of axioms for revisions, which we will not repeat here. The first main result of their theory is that these postulates for revisions are logically equivalent to the contraction postulates if the revision $A + z$ is defined by means of the *Levi identity* (after [Levi, 1977])

$$A + z \stackrel{\text{def}}{=} (A \dashv \neg z) + z, \quad (1)$$

so that revision by z is equivalent to contracting by $\neg z$ to remove any inconsistent beliefs and then expanding with z . One can also define contractions in terms of revisions by means of the *Harper identity* (after [Harper, 1976])

$$A \dashv z \stackrel{\text{def}}{=} (A + \neg z) \cap A, \quad (2)$$

so that the contraction by z is equivalent to taking those beliefs that would be preserved if $\neg z$ were now believed.

2.2 EPISTEMIC ENTRENCHMENT

Though the AGM axioms characterize rational revisions, Gärdenfors [1988] views the behaviors these axioms describe as arising from a more fundamental notion, that of *epistemic entrenchment*. Epistemic entrenchment is characterized by a complete preorder (a reflexive and transitive relation) over propositions which indicates which propositions are more valuable than others. This ordering, which may vary from belief state to belief state, influences revisions by the requirement that revisions retain more entrenched beliefs in preference to less entrenched ones.

If x and y are propositions, we write $x \leq y$ to mean that y is at least as epistemically entrenched as x . We define the strict part of this order, $x < y$, which means that y is more entrenched than x , by the conjunction of $x \leq y$ and $y \not\leq x$. The following axioms characterize the qualitative structure of this order.

- (≤ 1) If $x \leq y$ and $y \leq z$, then $x \leq z$; (*transitivity*)
- (≤ 2) If $x \vdash y$, then $x \leq y$; (*dominance*)
- (≤ 3) Either $x \leq z \wedge y$ or $y \leq z \wedge x$; (*conjunctiveness*)
- (≤ 4) If A is a consistent theory, then $x \leq y$ for all y iff $x \notin A$; (*minimality*)
- (≤ 5) If $x \leq y$ for all x , then $\vdash y$. (*maximality*)

Axiom (≤ 1) just says that \leq is an ordering relation, while the other axioms all concern how the logic of propositions interacts with the ordering. Postulate (≤ 2) says that x entails y , then retracting x is a smaller change than retracting y , since the closure requirement on belief states means that y cannot be retracted without giving up x as well. Axiom (≤ 3) reflects the fact that a conjunction cannot be retracted without giving up at least one of its conjuncts. Taken together axioms (≤ 1)-(≤ 3) imply that \leq is a complete ordering, that is, that either $x \leq y$ or $y \leq x$. Propositions not in a belief state are minimally entrenched in that state, according to (≤ 4), and according to (≤ 5), the only way a proposition can be maximally entrenched is if it is logically valid.

The influence of epistemic entrenchment on belief revisions is characterized by two conditions relating entrenchment orderings and contraction functions over theories. The first condition,

$$x \leq y \text{ iff either } x \notin A \dashv (z \wedge y) \text{ or } \vdash z \wedge y, \quad (3)$$

says that in contracting a theory A with respect to a conjunction, we must give up the conjunct of lesser epistemic entrenchment, or both conjuncts if they are equally entrenched. It says, in essence, that $x < y$ is the same as $y \in A \dashv (z \wedge y)$. The second condition,

$$y \in A \dashv z \text{ iff } y \in A \text{ and either } x < z \vee y \text{ or } \vdash z, \quad (4)$$

explicitly characterizes contraction functions in terms of epistemic entrenchment orderings.

The main result of the theory of epistemic entrenchment is that this notion is essentially equivalent to the previously axiomatised notion of rational revision of theories. Gärdenfors and Makinson [1988] prove that rational contraction functions may be constructed from orderings of epistemic entrenchment, and that entrenchment orderings may be constructed from rational contraction functions.

2.3 SOME IMPORTANT CONTRACTION FUNCTIONS

The central construct in many studies of contractions and revisions is the set of all maximal subsets of a belief state A consistent with a proposition x , which we write as $A \downarrow x$ and read as “ A less x .” Formally, we have

$$A \downarrow x \stackrel{\text{def}}{=} \{B \subseteq A \mid B \not\models x \wedge (B \subset C \subseteq A) \rightarrow C \models x\}.$$

It is easy to see that if A is a theory, so are the elements of $A \downarrow x$.

Using this construct, one can define the contraction $A \dashv x$ as the set of beliefs in either one, some, or all states in $A \downarrow x$. These definitions correspond to the notions of maxichoice, partial meet, and full meet contractions. We will consider only the first two of these here.

2.3.1 Maxichoice contraction

Maxichoice contraction is contraction to one maximal consistent subset. If there are no alternatives, then x must be logically valid and impossible to retract, so we may take A itself as the contraction. Formally, we assume the existence of a choice function C which selects one of the elements of $A \downarrow x$ and define the maxichoice contraction operation \sqsubseteq by

$$A \sqsubseteq x \stackrel{\text{def}}{=} \begin{cases} C(A \downarrow x) & \text{if } \nexists x \\ A & \text{otherwise.} \end{cases}$$

Applied to theories, this operation satisfies $(\perp 1)$ - $(\perp 6)$, but not necessarily $(\perp 7)$ and $(\perp 8)$ [Gärdenfors, 1988, Lemma 4.1]. Maxichoice contraction does satisfy $(\perp 7)$ and $(\perp 8)$ in the case that the choice function C is *orderly*, that is, if there is some partial ordering \sqsubseteq of all subsets of A such that C always chooses an element of $A \downarrow x$ that is maximal with respect to \sqsubseteq [Gärdenfors, 1988, Lemma 4.3]. However, failure to satisfy $(\perp 7)$ and $(\perp 8)$ is the least of the problems with maxichoice contraction, for one may prove that using maxichoice contraction to effect the revision $A + x$ via (1) makes $A + x$ a *complete* theory as long as $\neg x \in A$ [Gärdenfors, 1988, Corollary 4.6]. That is, $A + x$ in this case contains either y or $\neg y$ for each proposition y . Since A need not have contained either y or $\neg y$, these new beliefs are clearly gratuitous.

2.3.2 Partial meet contraction

The operation of *partial meet* contraction, $A \vdash x$, assumes a selection function S which selects subsets of $A \downarrow x$, and defines contraction by

$$A \vdash x \stackrel{\text{def}}{=} \begin{cases} \bigcap S(A \downarrow x) & \text{if } \nexists x \\ A & \text{otherwise.} \end{cases} \quad (5)$$

Partial meet contraction satisfies the basic rationality postulates $(\perp 1)$ - $(\perp 6)$, and in fact is equivalent to them in the sense that any operation satisfying these axioms is a partial meet contraction operation [Gärdenfors, 1988, Theorem 4.13].

Some partial meet contractions over theories, namely those derived from orderings of theories, are fully rational, satisfying $(\perp 1)$ - $(\perp 8)$. We say that the contraction operation is *relational* if there is a binary relation \sqsubseteq over 2^A such that the selected subsets are exactly the \sqsubseteq -“maximal” subsets, that is, if

$$S(A \downarrow x) = \{B \in A \downarrow x \mid \forall C \in (A \downarrow x) \quad C \sqsubseteq B\}. \quad (6)$$

This gives us a way of constructing a selection function, and hence a partial meet contraction relation, from every relation \sqsubseteq over $A \downarrow x$. The most important case is that when \sqsubseteq is a transitive relation. Gärdenfors [1988, Theorem 4.16] proves that if \sqsubseteq is transitive, the contraction function defined by (5) and (6) satisfies $(\perp 1)$ - $(\perp 8)$. He also proves that for each transitive relation \sqsubseteq there is a transitive total order \sqsubseteq' which yields the same contraction function as \sqsubseteq [Gärdenfors, 1988, Theorem 4.17]. That is, if a contraction function satisfies $(\perp 1)$ - $(\perp 8)$, there is a complete preordering of belief states such that contraction is partial meet contraction with respect to this total ordering.

3 REVISING BELIEF BASES

The AGM theory is not directly applicable to artificial intelligence since it requires that revisions of even finite belief states be infinite belief states. Nebel [1989] modified the AGM theory to yield finite revisions of belief bases. He defines belief base contraction, which we write as Θ , by

$$B \Theta x \stackrel{\text{def}}{=} \begin{cases} (B \vee \neg x) \wedge \bigvee_{C \in (B \downarrow x)} C & \text{if } \nexists x \\ B & \text{otherwise,} \end{cases}$$

and belief base revision (\oplus) by

$$B \oplus x \stackrel{\text{def}}{=} (B \Theta \neg x) \wedge x.$$

We may view Θ as an “implementation” of \dashv by lifting base revision to the belief state level. That is, we may use the contraction postulates to judge base contraction by identifying A with $Cn(B)$ and identifying $A \dashv x$ with $Cn(B \Theta x)$. With these identifications, Nebel [1989, Lemma 11] proves that belief base contraction satisfies $(\perp 1)$ - $(\perp 6)$.

Nebel proceeds to show that belief base contraction is a form of partial meet contraction when lifted to belief states. Let S_B be a selection function corresponding to theory B and defined by

$$\begin{aligned} S_B(Cn(B) \downarrow z) &\stackrel{\text{def}}{=} \\ &\{C \in (Cn(B) \downarrow z) \mid \forall C' \in (Cn(B) \downarrow z) \\ &C' \cap B \not\supseteq C \cap B\}. \end{aligned} \quad (7)$$

Nebel [1989, Theorem 14] then proves that contraction of finite premise sets B using Θ is identical (with respect to \vdash) to a partial meet contraction P_B defined by the selection function S_B , that is,

$$Cn(B \Theta z) = Cn(B) P_B z.$$

This guarantees that the lifted version of Θ satisfies (-1) - (-6) . By defining the order \sqsubseteq_B by

$$X \sqsubseteq_B Y \text{ iff } X \cap B \not\supseteq Y \cap B,$$

Nebel proves that partial meet contraction using S_B satisfies (-7) as well [Nebel, 1989, Theorem 15]. Since the order \sqsubseteq_B is not transitive in general, this definition of contraction need not satisfy (-8) .

These operations depend very strongly on the form of the belief base. In particular, iterated contraction of belief bases does not always make much sense, since Θ and \oplus sometimes replace all previous base propositions with a single new proposition that was not an element of the previous set. Subsequent contraction may then discard this singular residue of the original belief base.

While one may apply the notion of epistemic entrenchment in the foundational view of belief revision, this approach is not always practical. The dominance axiom (≤ 2), which requires that the logical consequences of a belief be at least as epistemically entrenched as the belief itself, means that to determine whether one proposition is more entrenched than another may require determining whether it entails the other, and since entailment is not decidable, this may not be possible. Nebel [1990, pp. 162-166] introduces the notion of *epistemic relevance* as an analogue of epistemic entrenchment more suited to the needs of computational belief revision. The basic idea is to view the syntactical form of the belief base as indicating or determining which propositions are *relevant* to the agent's purposes, and to view the propositions not in the belief base as irrelevant. The guideline for belief revision is then to minimize the loss of epistemically relevant propositions. Nebel proposes the selection function S_B defined in (7) as a formalization of this notion.

Nebel goes on to enlarge the conception of epistemic relevance from a simple binary distinction to a total preordering \leq , of the propositions in B , where $x \leq y$ means that y is at least as relevant as x . Unlike epistemic entrenchment orderings, epistemic relevance orderings may be arbitrary orderings of propositions,

regardless of any logical dependencies existing among the propositions. Nebel shows how such orderings of epistemic relevance can be used to embed \sqsubseteq_B in a transitive total ordering \sqsubseteq_ρ over subsets of B , defined by

$$X \sqsubseteq_\rho Y \text{ iff } \forall z \in (X - Y) \exists y \in (Y - X) \quad z \leq_\rho y.$$

This order ranks X less relevant than Y just in case X 's most relevant elements are less relevant than Y 's most relevant elements, ignoring the elements X and Y have in common. Since the propositions in B are totally ordered by \leq_ρ , \sqsubseteq_ρ totally orders all subsets of B . Furthermore, since over subsets of B it is clear that $X \sqsubseteq_B Y$ just means $X \subset Y$, we see that $X \sqsubseteq_\rho Y$ holds trivially if $X \sqsubseteq_B Y$, so the new order extends the old one. Nebel then defines a new selection function $S_{B,\leq}$, in terms of \sqsubseteq_ρ , by

$$\begin{aligned} S_{B,\leq}(A \downarrow z) &\stackrel{\text{def}}{=} \\ &\{C \in (A \downarrow z) \mid \forall C' \in (A \downarrow z) \quad C' \cap B \sqsubseteq_\rho C \cap B\} \end{aligned}$$

and proves [Nebel, 1989, Theorem 16] that partial meet contraction defined by this selection function gives rise to a fully rational contraction function satisfying (-1) - (-8) .

If \leq_ρ is a linear ordering (no ties allowed), then every finite set of beliefs has a most relevant element, and we may rewrite the definition of \sqsubseteq_ρ as

$$X \sqsubseteq_\rho Y \text{ iff } \max(X - Y) \leq_\rho \max(Y - X).$$

In this case, \sqsubseteq_ρ always singles out the greatest element of $(B \downarrow z)$, and the partial meet contraction function defined using $S_{B,\leq}$, resembles a maxichoice contraction on the belief base (see [Nebel, 1989, Lemma 17]). Nebel concludes from this that maxichoice contraction on belief bases does not have the undesirable completion behavior exhibited by maxichoice contraction of belief states. Moreover, maxichoice base contraction is just as rational as the more complicated base contraction. Since it can be iterated easily and does not introduce complicated disjunctions, it is to be preferred in practice.

4 CHOOSING REVISIONS RATIONALLY

The AGM principles of rational belief revision are intended to capture logical constraints on revisions and contractions, as opposed to pragmatic influences. Accordingly, the axioms for contraction functions concern mainly logical relationships among beliefs. But it is not clear that there are any constraints free of pragmatic motivation. In particular, the whole motivation for conserving as many beliefs as possible seems entirely nonlogical. Conservatism has nothing to do with the consistency or completeness of beliefs or the soundness of inferences, which are the *only* characteristics of

concern to logic. As Gärdenfors [1989] acknowledges, the motivations for conservatism are instead economic: beliefs are valuable (useful in acting, costly to infer or acquire), so getting rid of beliefs unnecessarily is irrational. This economic motivation for conservatism may not be pragmatic in the sense of being specific to some particular reasoning situation, but it is certainly not logical.

If we really desire a theory of rational belief revision, we must expand the notion of rationality from the purely logical sense, in which one is rational if one's beliefs are consistent and one's inferences are sound, to the economic sense, in which one is rational if one has a consistent set of preferences and makes choices that are optimal with respect to these preferences (cf. [Doyle, 1990]). This means the theory should use preferences about revisions in guiding revisions, and draw on the standard theory of economic rationality to formalize the notions of rational contraction and revision. In particular, the theory should take seriously the variability of the costs and benefits of beliefs. If the theory is to be general and cover all reasoning, it should allow preferences to vary with the reasoning situation and task, and should not presuppose any special measures of costs and benefits, but instead encompass all dimensions of value for guiding revisions.

According to the normative theory of economic rationality, rational agents choose maximally preferred alternatives. Preferences may stem from many different motivations, such as computational costs or moral principles, but all the formal theory requires is that a preference ordering \precsim is a complete, reflexive, and transitive relation that represents the agent's judgments of relative overall preferability (or utility) of possible alternatives. That is, \precsim satisfies, for all alternatives X , Y , and Z , the axioms

- (\precsim 1) Either $X \precsim Y$ or $Y \precsim X$, and (completeness)
- (\precsim 2) If $X \precsim Y$ and $Y \precsim Z$, then $X \precsim Z$. (transitivity)

It follows from axiom (\precsim 1) that the order \precsim is also reflexive, that is, $X \precsim X$. We use \prec to denote strict preference, the asymmetric part of the preference order, and \sim to indicate indifference, the reflexive part of the order. Thus $X \prec Y$ iff $X \precsim Y$ and $Y \not\precsim X$, and $X \sim Y$ iff $X \precsim Y$ and $Y \precsim X$. The combined order \precsim is sometimes called weak preference. The agent's preference ordering may change with its state, but we will consider only instantaneous preferences.

Belief revision involves moving from one belief state to another, so the choice in question to which preferences must apply is the choice of one belief state over alternative belief states. In practice, preferences among complete belief states will be constructed from preferences among classes among belief states. The simplest such preferences are those corresponding to preferences among individual beliefs. For example, we

may construct preferences among belief states from the ordering of propositions according to epistemic entrenchment. Intuitively, $x \leq y$ means that $X \sqsubseteq Y$ for any X and Y such that $x \in X$ and $y \in Y$. However, some important preference orders over belief states, such as the preferential interpretation of default rules proposed by Doyle and Wellman [1991], are more complex than comparisons of individual beliefs.

4.1 RATIONALITY AND SKEPTICISM

If we wish contraction based on epistemic entrenchment or epistemic relevance to be rational in the logical sense, then the resulting belief states must be maximal with respect to these orderings of belief states. That is not usually the case, however. First consider epistemic entrenchment. The maximally preferred states in $A \downarrow x$ are just the states in $S(A \downarrow x)$. But partial meet contraction makes the new belief state be the intersection of these. In the terminology of inheritance and default reasoning, partial meet contraction is skeptical [Horty et al., 1990]. The difficulty is that while skepticism is sometimes rational, the skepticism induced by partial meet contraction is rarely rational, since the intersection of preferred belief states is generally not itself a preferred belief state, that is, generally we have $\bigcap S(A \downarrow x) \notin S(A \downarrow x)$. For example, if we judge the preferability of belief states along numerous dimensions, each of the maximal belief states will be better along some dimensions and worse along others, and the intersection will be nonoptimal along as many dimensions as there are alternatives. The problem is clearest when examined in terms of propositional entrenchment. Consider the contraction $A \dashv (x \wedge y)$, and suppose that $x \leq y$, $\nvdash x$, $\nvdash y$, and $x, y \in A$. If $x < y$, then x is given up, while if $y \leq x$, both x and y are given up. There is no way to take the entrenchment-equivalence of x and y as a license to give up just one, chosen indifferently. Rather than increasing rationality, the skepticism resulting from choosing intersections of preferred belief states usually results in choosing suboptimal belief states. We conclude that if belief revision is to be rational, the belief states obtained in contraction and revision must be rational choices themselves rather than the intersection of rational choices.

4.2 RATIONALITY AND DETERMINISM

The first consequence of abandoning the skeptical approach to belief revision is that there is no longer any epistemological reason from logic or economics to suppose that contraction or revision are deterministic operations, as is presupposed in the AGM theory by viewing contraction and revision as functions taking belief states into belief states. Rationality does not prevent belief revision from being functional in deterministic agents. Even if the deterministic construction of the agent ensures that only one of the logically pos-

sible revisions is constitutionally possible, every revision performed may still be optimal with respect to the logically possible revisions. But determinism is in no sense a logical constraint on belief revision. This point is also made by Lindström and Rabinowics [1989], who develop a nondeterministic extension of the AGM theory.

Accordingly, we expand the formalization of contraction and revision to cover the cases in which contraction and revision are correspondences, that is, set-valued functions taking propositions and belief states into sets of belief states. We write $\hat{-}$ to denote rational contraction, and $\hat{+}$ to denote rational revision. For example, if \sqsubseteq x we might define the rational contraction $A \hat{-} x$ to be $S(A \downarrow x)$ if the selection function S is based on a preference order. Of course, we can always construct skeptical deterministic contraction and revision functions from nondeterministic ones by intersection, that is, by defining $A \hat{-} x$ to be $\bigcap(A \hat{-} x)$.

4.3 RATIONAL CONTRACTION AND REVISION

Even if \sqsubseteq represents preferences over belief states and $\bigcap S(A \downarrow x) \in S(A \downarrow x)$, so that the intersection of the preferred alternatives is itself a preferred alternative, partial meet contraction still need not be rational because there is no reason *a priori* why maximally preferred belief states should be sought only among the sets in $A \downarrow x$. Accordingly, we allow contraction to select among all belief states generated by subsets lacking x . The appropriate definitions vary depending on whether belief states are viewed as theories or as belief bases.

We define $A \Downarrow x$, read "A without x ," by

$$A \Downarrow x \stackrel{\text{def}}{=} \{X \subseteq A \mid X \not\sqsubseteq x\}.$$

Clearly, $A \downarrow x \subseteq A \Downarrow x$ for all A and x . We indicate the closures of these consistent subsets by $A \Downarrow^* x$, defined by

$$A \Downarrow^* x \stackrel{\text{def}}{=} \{Cn(X) \mid X \in A \Downarrow x\}.$$

We then define rational contraction of theories by

$$A \hat{-} x \stackrel{\text{def}}{=} \begin{cases} \{A\} & \text{if } \vdash x \\ \{A\} & \text{if } A \hat{-}' x = \emptyset \\ A \hat{-}' x & \text{otherwise} \end{cases} \quad (8)$$

where

$$A \hat{-}' x \stackrel{\text{def}}{=}$$

$$\{X \in (A \Downarrow^* x) \mid \forall X' \in (A \Downarrow^* x) \quad X' \not\preceq X\}.$$

Similarly, we define rational revision of theories by

$$A \hat{+} x \stackrel{\text{def}}{=} \begin{cases} \{L\} & \text{if } \vdash \neg x \\ \{L\} & \text{if } A \hat{+}' x = \emptyset \\ A \hat{+}' x & \text{otherwise} \end{cases} \quad (9)$$

where

$$A \hat{+}' x \stackrel{\text{def}}{=}$$

$$\{X + x \mid X \in (A \Downarrow^* \neg x) \wedge \forall X' \in (A \Downarrow^* \neg x) \quad X' + x \not\preceq X + x\}.$$

Although we will study the implications of these definitions in the following, they should only be taken as initial attempts to define the notions of rational contraction and rational revision. The reader will note, for example, that the cases defining contraction and revision in the cases $\vdash x$ and $\vdash \neg x$, respectively, are each subsumed by the immediately following conditions in (8) and (9). The former cases correspond to there being no possible effective contraction or revision, while the latter cases correspond to there being possible effective contractions or revision, but no maximally preferable ones. The definitions make sense in spite of the subsumption because the same result is indicated for both of these cases. But it is not at all clear that the two deserve to be treated the same way.

There are even more interesting possibilities for defining rational contraction and revision of belief bases. For example, we may choose contractions and revisions either by ranking belief bases themselves (comparing them directly), or by ranking their closures (judging them by their "effects" as it were). Moreover, the contraction and revision operations may be permitted to perform some amount of deductive inference, that is, to augment the belief bases with parts of their closures, or may be restricted to sets of beliefs deemed to be "legal" states. Space limitations preclude formal elaboration of any of these possibilities.

The generality exhibited in the definitions of rational contraction and revision goes against the usual presupposition of epistemologists that knowing more is always better. Indeed, one can even prove that knowing more is better in some standard theories (see [Good, 1983]). But this presupposition (or theorem) is not always justified when holding or revising beliefs have non-epistemological effects, such as incurring computational costs that must be borne by the agent. For example, if it takes too long to compute enough of a revision that satisfies the AGM postulates, a practical belief revision method might instead simply seek to remove the easily identified inconsistencies between the belief state and the new information, rather than work to remove all inconsistencies between these. In other cases, knowing some specific beliefs may make the agent worse off than if it did not hold them. For example, 1984's Winston Smith is much better off not knowing some of the facts about his crazy world, since he will be severely tortured if he reveals this knowledge, and has no hope of keeping up a perfect pretense of ignorance. More mundanely, there are many facts that people find very painful to know (infidelity of a spouse, criminality of an employer) unless they also know enough extra facts to permit effective action (ad-

equate legal grounds for divorce, sufficient evidence for conviction). The general pattern here is that for certain questions believing something may be worse than not believing it, even though not believing it is in turn worse than believing it for good reason.

4.4 ECONOMIC VS. AGM RATIONALITY

Do economically rational contractions satisfy the AGM postulates? The obvious answer is no: rational contraction is of a different type than AGM contraction, a nondeterministic relation rather than a functional relation. Nevertheless, we can compare rational contractions with AGM contractions in a more interesting way by seeing if deterministic contraction operations that are rational in the economic sense need also satisfy the AGM axioms. Formally, we say that $\dot{-}$ is a ratichoice contraction if it is both rational and functional, that is, if $A \dot{-} z \in A \dot{-} z$ for every A and z . We then have the following result.

Theorem 1 *Ratichoice contraction of theories satisfies the AGM axioms $(\dot{-}1)$, $(\dot{-}2)$, and $(\dot{-}4)$, and can violate $(\dot{-}3)$, $(\dot{-}5)$ - $(\dot{-}8)$, and the Levi and Harper identities (1) and (2) .*

Proof: Suppose that $\dot{-}$ is a functional rational contraction operation over closed belief states and that x , y , and z are logically independent and collectively consistent propositions.

To begin with, $A \dot{-} z$ is clearly a theory, so $(\dot{-}1)$ holds. Similarly, $(A \dot{-} z) \subseteq A$, so $(\dot{-}2)$ holds, as does $(\dot{-}4)$.

We show that the remaining axioms need not hold by exhibiting preference orders that demonstrate this. First, though, we introduce some notation to make describing the preference orders more convenient. If \dot{X} and \dot{Y} are disjoint sets of subtheories of A , we write $\dot{X} \triangleleft \dot{Y}$ to mean that $X \prec Y$, $X \sim X'$ and $Y \sim Y'$ for every $X, X' \in \dot{X}$ and $Y, Y' \in \dot{Y}$. We also write H_z (read "holds z ") as an abbreviation for the subtheories of A in which the proposition z is held, that is, $H_z \stackrel{\text{def}}{=} \{Cn(X) \subseteq A \mid z \in Cn(X)\}$. Similarly, $\neg H_z$ means the complement of H_z , and $H_z \wedge H_w$ (resp. \vee) means the intersection (union) of these subsets.

Now suppose that $A = Cn(\{y\})$ and $A \not\vdash z$, so $A \Downarrow z = \{Cn(\emptyset), Cn(\{y\}), Cn(\{x \vee y\}), \dots\}$. If $H_y \triangleleft \neg H_y$, then $A \dot{-} z \neq A$, so $(\dot{-}3)$ need not be satisfied. Similarly, if $A = Cn(\{x, y\})$ and $H_y \triangleleft \neg H_y$, it may be that $A \dot{-} z = Cn(\emptyset)$, in which case $(X \dot{-} z) + z = Cn(\{x\})$ and $A \not\subseteq Cn(\{x\})$. Thus $(\dot{-}6)$ need not hold.

Suppose that $\vdash z \leftrightarrow y$. It is clear that the nondeterministic analogue of the equivalence axiom $(\dot{-}5)$ holds for rational contraction, in that $A \dot{-} z = A \dot{-} y$. But since a ratichoice contraction may select a different alternative from $A \dot{-} z$ when contracting by z than when contracting by y , $(\dot{-}5)$ need not hold.

Next, neither $(\dot{-}7)$ nor $(\dot{-}8)$ always hold. Suppose that $A = Cn(\{x, y, z\})$. If

$$\begin{aligned} H_x \wedge \neg H_y \wedge \neg H_z &\triangleleft \neg H_x \wedge H_y \wedge \neg H_z \\ &\triangleleft \neg H_x \wedge \neg H_y \wedge H_z \triangleleft \neg H_x \wedge \neg H_y \wedge \neg H_z \\ &\triangleleft H_x \wedge \neg H_y \wedge H_z \triangleleft \neg H_x \wedge H_y \wedge H_z, \end{aligned}$$

then $Cn(\{y, z\})$ is maximal in $A \Downarrow^* z$, $Cn(\{x, z\})$ is maximal in $A \Downarrow^* y$, and $Cn(\emptyset)$ is maximal in $A \Downarrow^* (x \wedge y)$, in which case $(\dot{-}7)$ may be violated because

$$Cn(\{y, z\}) \cap Cn(\{x, z\}) = Cn(\{z\}) \neq Cn(\emptyset).$$

On the other hand, if

$$\begin{aligned} \neg H_x \wedge \neg H_y \wedge \neg H_z &\triangleleft H_x \wedge \neg H_y \wedge H_z \\ &\triangleleft \neg H_x \wedge H_y \wedge H_z \triangleleft \neg H_x \wedge \neg H_y \wedge H_z \\ &\triangleleft H_x \wedge \neg H_y \wedge \neg H_z \triangleleft \neg H_x \wedge H_y \wedge \neg H_z, \end{aligned}$$

then $Cn(\{y\})$ is maximal in $A \Downarrow^* z$ and $Cn(\{z\})$ is maximal in $A \Downarrow^* (x \wedge y)$, in which case $(\dot{-}8)$ may be violated because $Cn(\{z\}) \not\subseteq Cn(\{y\})$.

Finally, neither the Levi nor Harper identities always hold. Suppose $A = Cn(\{\neg x, y\})$. If

$$\neg H_x \wedge H_y \triangleleft \neg H_x \wedge \neg H_y \triangleleft H_x \wedge \neg H_y \triangleleft H_x \wedge H_y,$$

it may be that $A \dot{-} \neg x = Cn(\emptyset)$ and $A \dot{+} x = \{Cn(\{x, y\})\}$, in which case

$$(A \dot{-} \neg x) + z = \{Cn(\{x\})\} \neq A \dot{+} z,$$

which contradicts the Levi identity (1) . Similarly, we have

$$(A \dot{+} z) \cap A = Cn(\{y\}) \neq A \dot{-} \neg x,$$

which contradicts (with x and $\neg x$ interchanged) the Harper identity (2) . \square

These complications may be avoided if we require that contractions be chosen from among $A \downarrow z$ rather than $A \Downarrow^* z$, as shown by the following result.

Theorem 2 *If $(A \dot{-} z) \subseteq (A \downarrow z)$ whenever $\not\vdash z$, then ratichoice contraction satisfies the AGM axioms $(\dot{-}1)$ - $(\dot{-}8)$.*

Proof: Suppose $(A \dot{-} z) \subseteq (A \downarrow z)$ whenever $\not\vdash z$. Then there is some maxichoice function C such that $C(A \downarrow z) \in (A \dot{-} z)$ for all z such that $\not\vdash z$. Since each particular contraction given by $\dot{-}$ can be viewed as the result of a maxichoice contraction, ratichoice contraction in this case satisfies all the axioms that maxichoice contraction satisfies, namely (1) , (2) , and $(\dot{-}1)$ - $(\dot{-}6)$. In fact, ratichoice contraction satisfies $(\dot{-}7)$ and $(\dot{-}8)$ as well because any ratichoice contraction operation is *orderly*, that is, any element of $A \dot{-} z$ is an element of $A \downarrow z$ that is maximal with respect to a partial ordering of subsets of A . As mentioned earlier, any orderly maxichoice contraction operation satisfies all the contraction axioms, and to apply this result, we need only

consider the partial order \sqsubseteq defined so that $X \sqsubseteq Y$ iff either $X = Y$ or $X \prec Y$. \square

Of course, the conditions of Theorem 2 leads to ratichoice contractions that share the undesirable properties of maxichoice contraction of theories noted earlier: if $\neg z \in A$, then every proposition is either believed or denied when one first contracts with respect to $\neg z$ and then extends by z . It appears, however that the ratichoice revision by z in the same circumstances need not be complete, since it need not satisfy the Levi identity, but I have no proof of this.

Another way of avoiding the complications of ratichoice contraction is to require that believing more is better than believing less. Formally, we say that the preference relation \precsim is (*positively*) *informationally monotone* if $X \precsim Y$ whenever X and Y are both consistent and $Y \vdash X$. We then have the following result.

Theorem 3 *If \precsim is informationally monotone, then ratichoice contraction satisfies the Levi and Harper identities and the AGM axioms (–1)–(–8).*

Proof: Obviously, if \precsim is informationally monotone, then $(A \hat{\sim} z) \subseteq (A \downarrow z)$ if $\nvdash z$, so Theorem 2 applies. \square

Many questions remain to be answered. For example, are the AGM axioms equivalent to specific conditions on preferences over belief states? Do rational state preferences induce rational propositional preferences?

5 PARTIAL EPISTEMIC PREFERENCES

Nebel suggests two reasons why epistemic relevance is a more practical basis for belief revision than epistemic entrenchment. In the first place, epistemic relevance orderings need not respect logical dependencies among propositions. That is, we are free to order $x \leq, y$ without regard to whether $x \vdash y$ or $y \vdash x$, in contrast with the dominance condition (≤ 2) on epistemic entrenchment orderings, which can be quite costly (or impossible) to ensure. In the second place, linear epistemic relevance orderings of the propositions in a belief base make belief base contraction easy to implement by simply dropping the lowest ranked propositions in any conflicting set (as in RUP [McAllester, 1982]). Moreover, this form of belief base contraction is fully rational, and corresponds to maxichoice contraction on belief bases [Nebel, 1989, Lemma 17ff]. Representing epistemic entrenchment, however, can be more costly. Gärdenfors and Makinson [1988] show that epistemic entrenchment orders can be represented by information linear in the size of a certain algebraic construct from belief states (the dual atoms of the lattice of equivalence classes of beliefs). Unfortunately, this may

still be exponential in the number of atomic propositions in the belief base. But pure representational size is only half the problem, and they leave the problem of logical dependencies unaddressed. Orderings of propositions and belief bases will only be useful in practice if they can be both represented and computed quickly.

While Nebel's epistemic relevance orderings make rational belief revision more practical, this approach is not without significant costs. The main problem is the inflexibility of this means for effecting contractions. Specifically, linear epistemic relevance orderings rank each possible contraction by the most valuable proposition retained, irrespective of what other propositions are retained. This might seem reasonable since \sqsubseteq_ρ is applied to the elements of $B \downarrow z$, in which case choosing the subset with the maximal element is the same as choosing the subset which abandons the least valuable propositions. But suppose $X = \{a_1, \dots, a_{1000}\}$ and $Y = \{a_{1001}\}$, where \leq_ρ ranks these propositions by the natural indicial order, so that $X \sqsubseteq_\rho Y$. If all of these propositions are of roughly the same value (but each differing slightly from the rest), then the revision

$$\{a_1, \dots, a_{1001}\} \ominus (a_{1001} \wedge (a_1 \vee \dots \vee a_{1000})) = Y$$

chooses among the two alternatives X and Y and discards a thousand good propositions in favor of a single proposition that is little better than those discarded. This seems unreasonable compared with, for example, using a weighted comparison or voting scheme in which equally valuable propositions get equal say in the selection. Unfortunately, there is no way to express schemes like majority voting with linear orders over propositions. More generally, achieving any dependence of ordering on the global composition of the alternatives means revising the linear propositional order to fit each set of alternatives.

It would be valuable to have some more flexible way of specifying preferences for guiding contraction and revision. If we look to the usual explanations of why one revision is selected over another, we see that many different properties of propositions influence whether one proposition is preferred to another. For example, one belief might be preferred to another because it is more specific, or was adopted more recently, or has longer standing (was adopted less recently), or has higher probability of being true, or comes from a source of higher authority. These criteria, however, are often partial, that is, each may be viewed as a preorder \precsim such that both $X \not\precsim Y$ and $Y \not\precsim X$ for some X and Y . For example, there are many different dimensions of specificity, and two beliefs may be such that neither is more specific than the other. Similarly, probabilities need not be known for all propositions, and authorities need not address all questions. Moreover, none of these are comprehensive criteria that take all possible considerations into account. If we want contraction and revision to be truly flexible, we need some way of combining different partial, noncomprehensive order-

ings of propositions into complete global orderings of belief states.

But combining partial orderings into a global ordering can be difficult because the partial criteria may conflict in some cases. To borrow an example from non-monotonic logic, we might reasonably prefer to believe that Quakers are pacifist, and that Republicans are not pacifists. These preferences can conflict on cases like that of Nixon, and a preference for more specific rules does not help since "Quaker" and "Republican" are incomparable categories. Indeed, as argued elsewhere [Doyle and Wellman, 1991], other preference criteria can conflict as well, including very specific criteria corresponding to individual default rules. Constructing a global ordering thus means resolving the conflicts among the preference criteria being combined.

In addition to flexibility, we seek a revision method which is potentially mechanizable. This means that whatever method is employed for resolving conflicts must also be mechanizable because placing responsibility for resolving potential conflicts on the theorist is infeasible. For large sets of criteria it is difficult to anticipate all of the potential conflicts and all of the varying circumstances that may influence how the conflicts should be resolved. It also seems difficult to anticipate discovery of new criteria. Thus we seek conflict resolution mechanisms based on general, modular rules of combination that apply even as the criteria used evolve.

5.1 CONSTRUCTING GLOBAL PREFERENCE ORDERS

To analyze the problem of modular construction of orderings, we follow the formal approach elaborated by Doyle and Wellman [1991] for analyzing the related problem for preference-based nonmonotonic logics. We say that an *aggregation policy* is a function that specifies the global order corresponding to any given set of partial preference orders. Let the set I index the set of partial preference orders that are to be combined, so that if $i \in I$, \preceq_i denotes the preference order corresponding to the i th criterion for belief revision to be included in the overall revision conception. The problem is then to aggregate the set of orders $\{\preceq_i; i \in I\}$ into a global preference order \preceq .

The principled design of an aggregation policy for partial preference criteria begins with a consideration of properties we think a reasonable policy should exhibit. The properties we propose are analogs of Arrow's [1963] desiderata for social choice. (See [Doyle and Wellman, 1991] for further explanation and justification of these desiderata as principles for reasoning.)

1. *Collective rationality.* The global order \preceq is a function of the individual orders \preceq_i , which are unrestricted, possibly partial, preorders.

2. *Pareto principle (unanimity).* If $X \prec_i Y$ for some $i \in I$ and for no $j \in I$ does $Y \prec_j X$, then $X \prec Y$. In other words, the global order agrees with unanimous strict preferences.
3. *Independence of irrelevant alternatives (IIA).* The relation of X and Y according to the global order depends only on how the individual orders rank those two candidates. That is, considering new alternatives does not alter rankings among the originals.
4. *Nondictatorship (noncomprehensive criteria).* There is no $i \in I$ such that for every X and Y , $X \preceq Y$ whenever $X \preceq_i Y$, regardless of the \preceq_j for $j \neq i$. That is, there is no "dictator" whose preferences automatically determine the group's, no matter how the other individual orderings are varied.
5. *Conflict resolution.* If $X \preceq_i Y$ for some i , then $X \preceq Y$ or $Y \preceq X$. That is, if two candidates are comparable in an individual order, then they are comparable in the global order.

Leaving aside the conflict resolution condition for now, the following theorem states that the desirable and apparently reasonable properties enumerated above are not simultaneously satisfiable by any aggregation policy for preferences expressed by total orders.

Theorem 4 (Arrow) *If the domain includes more than two alternatives, no aggregation policy mapping sets of total preorders to total global preorders satisfies the collective rationality, Pareto, IIA, and nondictatorship conditions.*

We omit the proof because, with the restriction to total orders, this is exactly Arrow's theorem [Arrow, 1963].

There is no problem finding good aggregation policies for choices among only two alternatives: majority rule works fine, for example. But for the case of belief revision, there are typically several possible alternatives to choose from. This means that the following theorem applies to the typical case of belief revision. We omit the proof, which may be found, together with further discussion, in [Doyle and Wellman, 1991, Theorem 3].

Theorem 5 (Doyle-Wellman) *If the domain includes more than two alternatives, no aggregation policy for partial preference preorders satisfies the collective rationality, Pareto, IIA, nondictatorship, and conflict resolution conditions.*

Thus if rational belief revision requires a preorder that completely orders the alternative contractions or revisions, we may expect that the only general way of obtaining the order is manual construction, that is, to supply a dictatorial policy. One way to do this is to impose a linear ordering over all the criteria, so that the first criterion always gets its way regardless of what

the rest of the criteria say, unless it expresses no preference, in which case the second criterion gets its way, and so on. Choice rules of this form are called *lexicographic* because they resemble the method for ordering words alphabetically: compare the first letters; if tied compare the second, etc. But like Nebel's orderings, these linear orderings are not very flexible, and can be expected to require ongoing manual revision to achieve satisfactory performance.

There are a number of possible ways around Theorem 5, some of which are discussed in [Doyle and Wellman, 1991] in the context of nonmonotonic logic. We mention only one here: the approach of restricting the domain over which preference aggregation occurs. The collective rationality condition stipulates that the aggregation policy must work no matter what preference orders are presented for combination. But if one can show that all preference orderings of interest take one of several particular forms, these limitations may permit construction of an aggregation policy satisfying all of the conditions over this restricted domain of preferences. Does Theorem 5 still hold if each preference order satisfies properties like informational monotonicity or $Z \preceq X \cap Y$ whenever $Z \preceq X$ and $Z \preceq Y$? Are such restrictions on the allowable preferences reasonable outside the special domain of epistemology?

5.2 REVISION RATIONALITY AND AGGREGATION RATIONALITY

Some irrationalities in preference aggregation do not affect the formal rationality of rational contraction and revision, which requires only that choices be maximally preferable according to a total preference ordering. For example, aggregation policies which yield total preference orderings and so provide the necessary basis for applying (8) and (9) may nevertheless violate the Pareto, IIA, or nondictatorship principles.

Violations of the Pareto or nondictatorship principles just mean that the resulting belief revisions may not be rational with respect to the ignored preference criteria, even though they are maximally preferred with respect to the aggregate order. An aggregation policy that violates IIA, on the other hand, can exhibit erratic behavior. Specifically, it can lead to different results depending on the presence or absence of irrelevant information. Suppose, for example, that $A = Cn(\{x, y, z\})$, and that P_1 and P_2 are two preference criteria to be combined with P_1 satisfying

$$\neg Hz \wedge \neg Hy \wedge \neg Hx \lhd Hz \wedge \neg Hy \lhd \neg Hz \wedge Hy$$

and P_2 satisfying

$$\neg Hz \wedge \neg Hy \wedge \neg Hx \lhd \neg Hz \wedge Hy \lhd Hz \wedge \neg Hy$$

If the aggregation policy violates IIA, we may have a ratichoice contraction \dashv such that $A \dashv (x \wedge y) = \{Cn(\{y\})\}$ but $A \dashv (x \wedge y) \vee z = \{Cn(\{x\})\}$, with the

overall choice between $Cn(\{x\})$ and $Cn(\{y\})$ depending on what other alternatives are available.

In contrast to these formally rational but substantively irrational revision orderings, violations of collective rationality and the incompleteness addressed by conflict resolution can lead to global preference "orderings" that are intransitive or incomplete. This raises difficulties when such comparisons used to select among alternative contractions and revisions. We examine these difficulties in turn.

An aggregation policy may violate collective rationality in several ways. First, it may not aggregate every possible partial preference criterion. As with violations of the Pareto or nondictatorship conditions, this need not affect the formal rationality of belief revision. Second, the result of aggregation may not be transitive. The effects of this depend on just what we mean by intransitivity. Intransitivity of \preceq means that for some X, Y, Z we have $X \preceq Y$ and $Y \preceq Z$ but $X \not\preceq Z$. That is, if $A \Downarrow^* z = \{X, Z\}$, then both alternatives are maximally preferred, so either is a "rational" choice. On the other hand, intransitivity of \prec means that for some X, Y, Z we have $X \prec Y$, $Y \prec Z$, and $Z \prec X$. In this case, if $A \Downarrow^* z = \{X, Y, Z\}$, then no alternative is maximally preferred, so $A \dashv z = A$.

The definitions of rational contraction and revision presumed a complete preference ordering of belief states. But there is no objection in principle to weakening the definitions by allowing \preceq to be partial. (In this case, of course, we can no longer view $X \prec Y$ and $Y \not\preceq X$ as equivalent.) The effect of incompleteness is just to allow more alternatives to be maximally preferred than might be the case were the order to be total. Incompleteness when no relevant preferences are available does not seem particularly objectionable. But incompleteness when preferences do exist is a different matter, and such incompletions can arise either through intransitivity (as noted above), through aggregation policies that do not aggregate some preference orders, or through violations of the conflict resolution condition. In these cases, the incompleteness arises through an explicit violation of a reasonable rationality condition.

6 CONCLUSION

We examined the AGM theory of rational belief revision and its extension by Nebel. The heart of the AGM theory is the ordering of beliefs according to epistemic entrenchment and its equivalence with his notion of rational belief revision. Nebel adapted this theory to cover revision of finite bases of belief, and his results show that finite revision can be rational in AGM sense, and that in the finite case maxichoice revision is as rational as skeptical revision.

We argued that these theories are inadequate because

they demand more information about which revisions are preferable than is typically available in practice, and because their notion of "rationality" has little to do with the economic notion of rational choice among alternatives. We presented a modification of these theories which makes weaker, more realistic assumptions, namely that belief revisions must be guided by partial preferences which may conflict with each other. This theory is closely related to our theory of rational default inference, and supports similar results. We showed how the notion of rationality proposed by Alchourrón, Gärdenfors, and Makinson is violated by revisions which are rational in the economic sense, and proved that fully rational belief revision is sometimes impossible when rationality is judged by partial preferences.

It is an open problem to find rationality postulates to describe revision that is rational in the economic sense. There may not be any, though a complete axiomatization seems more likely if one presupposes a global expected utility preordering of beliefs or of belief states. There certainly will not be axioms that refer only to beliefs, as in the AGM axioms. In addition, Theorem 5 suggests that there may not be a single ideal theory of rational revision.

Acknowledgments

This work draws substantially on previous work conducted in collaboration with Michael Wellman, and I thank him for many valuable suggestions and for reading drafts of the full version of this paper. The paper has also been improved significantly through discussions with Peter Gärdenfors, David Makinson, and Bernhard Nebel, who each supplied many useful comments and pointed out errors in the preliminary version presented at the Third International Workshop on Nonmonotonic Reasoning. I also thank Sten Lindström, Włodzimierz Rabinowicz, Joseph Schatz, and Richmond Thomason for valuable discussions. This work was supported by the National Library of Medicine through National Institutes of Health Grant No. R01 LM04493 and by the International Business Machines Corporation through a grant to MIT's Laboratory for Computer Science.

References

- [Alchourrón et al., 1985] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction functions and their associated revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.
- [Arrow, 1963] K. J. Arrow. *Social Choice and Individual Values*. Yale University Press, 2nd edition, 1963.
- [Doyle and Wellman, 1991] J. Doyle and M. P. Wellman. Impediments to universal preference-based de-
- fault theories. *Artificial Intelligence*, 1991. To appear.
- [Doyle, 1990] J. Doyle. Rationality and its roles in reasoning (extended abstract). In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 1093–1100, Menlo Park, CA, 1990. AAAI Press.
- [Gärdenfors and Makinson, 1988] P. Gärdenfors and D. Makinson. Revisions of knowledge systems using epistemic entrenchment. In M. Y. Vardi, ed., *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–95, Los Altos, CA, March 1988. Morgan Kaufmann.
- [Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge, MA, 1988.
- [Gärdenfors, 1989] P. Gärdenfors. The dynamics of belief systems: Foundations vs. coherence theories. *Revue Internationale de Philosophie*, 1989.
- [Good, 1983] I. J. Good. On the principle of total evidence. In *Good Thinking: The Foundations of Probability and Its Applications*. University of Minnesota Press, 1983. Originally appeared in *British Journal of Philosophy of Science*, 17:319–321, 1967.
- [Harper, 1976] W. L. Harper. Rational conceptual change. In *PSA 1976*, volume 2, pages 462–494, East Lansing, MI, 1976. Philosophy of Science Association.
- [Horty et al., 1990] J. F. Horty, R. H. Thomason, and D. S. Touretzky. A skeptical theory of inheritance in nonmonotonic semantic networks. *Artificial Intelligence*, 42(2-3):311–348, March 1990.
- [Levi, 1977] I. Levi. Subjunctives, dispositions and chances. *Synthese*, 34:423–455, 1977.
- [Lindström and Rabinowicz, 1989] S. Lindström and W. Rabinowicz. Belief revision, epistemic conditionals and the Ramsey test. Department of Philosophy, Uppsala University, Sweden, 1989.
- [Makinson, 1987] D. Makinson. On the status of the postulate of recovery in the logic of theory change. *Journal of Philosophical Logic*, 16:383–394, 1987.
- [McAllester, 1982] D. A. McAllester. Reasoning Utility Package user's manual. AIM 667, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA, 1982.
- [Nebel, 1989] B. Nebel. A knowledge level analysis of belief revision. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 301–311, San Mateo, CA, 1989. Morgan Kaufmann.
- [Nebel, 1990] B. Nebel. *Representation and Reasoning in Hybrid Representation Systems*. Springer-Verlag, Berlin, 1990.

Conditional Objects and Non-Monotonic Reasoning

Didier Dubois - Henri Prade

Institut de Recherche en Informatique de Toulouse

Université Paul Sabatier, 118 route de Narbonne, 31062 Toulouse Cedex – France

Abstract

This paper pursues an investigation in the relationship between numerical approaches to uncertainty handling in Artificial Intelligence, and the concept of non-monotonic reasoning. The basic building block is the so-called «conditional object», obtained as a qualitative counterpart to conditional probability. Viewed as an inference rule, the conditional object is shown to possess almost all properties of a well-behaved non-monotonic consequence relation ; however the reflexivity property must be weakened. The calculus of conditional objects is proved to be consistent with non-standard probabilistic semantics of non-monotonic reasoning as well as with possibilistic logic. Both provide models for a well-behaved non-monotonic logic. Especially, it turns out that Geffner's qualitative probabilistic default logic based on Adams' semantics is equivalent to a form of possibilistic logic, and owes nothing to the additivity axiom of probability.

1 INTRODUCTION

Purely symbolic approaches as well as numerical models for reasoning under incomplete or uncertain knowledge have insisted on the importance of the correct handling of the context in which the inference is made; see Léa Sombé(1990) for a comparative overview of symbolic and numerical approaches. Among others in the symbolic school, Reiter and Crisculo (1981), Touretzky (1984) have quite early considered the problems raised by the simultaneous presence in the knowledge base of default rules whose condition parts have various level of specificity (e.g. "if a then c generally" and "if b then $\neg c$ generally" while a logically entails b). Using a different terminology, Kyburg (1974) has based his reasoning strategy on the choice of the right "reference class" (which is not always, according to him, the most specific one on which we have some information available), while Bayesian probability supporters emphasize the fact that a probability is always a conditional probability in order to take into account the current context (see Cheeseman

(1988), Pearl (1988) for instance). Indeed the values of $\text{Prob}(c|a)$ and of $\text{Prob}(c|b)$ remain unrelated in general even when a logically entails b . Thus the comparative study of conditioning in probability theory (or in other numerical models of uncertainty) with the weak consequence relations used in non-monotonic logics seems to be a natural and worth-considering question to investigate. In this paper, this is done by taking advantage of the notion of conditional object.

The idea of a conditional object corresponds to an attempt to give a mathematical and logical meaning to the entity $b|a$ independently of the notion of probability, but still in agreement with this notion in the sense that $\text{Prob}(b|a)$ can indeed be considered as the probability of the entity b (and not only as the probability of b , in the context a). This kind of study was pioneered by Schay(1968). There has been several recently published works along this research line, especially Calabrese (1987), Goodman and Nguyen (1988), Goodman, Nguyen and Walker (1991), Dubois and Prade (1988a, 1990a, 1991). In this paper we develop preliminary remarks made in (Dubois and Prade, 1989), and mainly focus our attention on the relationship between the logical calculus developed on conditional objects like $b|a$ and non-monotonic reasoning systems based on weak consequence relations like $a \sim b$ as studied by Gabbay (1985) and more recently by Kraus et al. (1990).

In the next section, the necessary background on conditional objects is restated in the framework of finite Boolean algebras. Section 3 is devoted to the comparative study of conditional objects and of weak consequence relationship in the sense of non-monotonic logic. Section 4 examines the agreement of probability theory and possibility theory with non-monotonic logic and conditional objects, and proposes a possibilistic interpretation of Adams's-e-semantics.

2 CONDITIONAL OBJECTS : BASIC RESULTS

2.1 DEFINITIONS AND TRUTH-TABLE INTERPRETATION

In (Dubois and Prade, 1988a), a conditional object $b|a$ linking two propositions a and b of propositional logic, has been given a semantics under the form of an

incomplete truth-table, namely denoting t the truth-assignment function, $t(b|a)=1$ when a and b are both true, $t(b|a)=0$ when b is false and a is true. When a is false, $b|a$ is considered as undefined, which is denoted as $t(b|a)=?$. Intuitively the conditional object $b|a$ is similar to the production rule "if a then b " in the sense that the rule applies only when a is true and then the conclusion ' b is true' can be produced provided that the rule is considered as true. The following truth-table exhibits the difference between the material implication $a \rightarrow b$ and the conditional object $b|a$

$t(a)$	$t(b)$	$t(a \rightarrow b)$	$t(b a)$
1	1	1	1
1	0	0	0
0	1	1	?
0	0	1	?

Thus the conditional object $b|a$ is understood as "in the context where a is true, b is regarded as true, in the context where a is false nothing can be said about b ". Besides the truth-value $t(b|a)$ can be also implicitly defined by means of the equation

$$t(a \wedge b) = t(b|a) * t(a) \quad (1)$$

where $*$ is the conjunction operation on $\{0,1\}$, i.e. $*$ is such that $1 * 1 = 1$ and $0 * x = x * 0 = 0$. Then any proposition x such that $t(a \wedge b) = t(x) * t(a)$ can stand for $b|a$. Thus the conditional object $b|a$ can be identified to a set of propositions, for instance $\{a \wedge b, b, a \leftrightarrow b, a \rightarrow b\}$ in a language containing only a and b as propositional symbols ; this corresponds to the four manners to substitute 0 or 1 to the two '?' in the above truth-table. Independently Goodman and Nguyen (1988) have proposed the following definition, where a, b, x denote elements of a Boolean algebra \mathcal{B} , $(\mathcal{B}, \wedge, \vee, \neg)$, supposed to be finite here for simplicity.

$$\forall a, b \in \mathcal{B}, b|a = \{x, x \wedge a = a \wedge b\}. \quad (2)$$

It is easy to verify that (1) corresponds to the semantics of the above definition. Let us observe that the introduction of a third truth-value together with equation (1) whose solutions define $t(b|a)$ lead to postulate $? * 0 = 0$.

Let \leq denote the usual partial ordering in the Boolean algebra \mathcal{B} , expressing entailment. So $a \leq b$ if and only if $\neg a \vee b = T$, where T denotes the greatest element in \mathcal{B} . Then we have

$$b|a = \{x, a \wedge b \leq x \leq \neg a \vee b\}. \quad (3)$$

Considering a Boolean algebra \mathcal{B} of propositions with tautology T and contradiction \perp , we can define the set $\mathcal{B} \setminus \mathcal{B} = \{b|a, (a, b) \in \mathcal{B}^2\}$ of conditional objects. \mathcal{B} can be

identified as the subset $\{a|T, a \in \mathcal{B}\}$; indeed $t(a|T) = t(a)$ from (1).

2.2 EQUALITY AND PARTIAL ORDERING IN $\mathcal{B} \setminus \mathcal{B}$

It can be checked, using (2), that

$$b|a = d|c \text{ if and only if } a \wedge b = c \wedge d \text{ and } a = c \quad (4)$$

Thus two conditional objects $b|a$ and $d|c$ are equal if and only if they have the same condition part (i.e. $a=c$) and their conclusion parts are identical when the condition is satisfied (i.e. $a \wedge b = c \wedge d$). The following equalities which are in agreement with our interpretation of $b|a$ are easy to check

$$b|a = (a \wedge b)|a = (a \leftrightarrow b)|a = (a \rightarrow b)|a \quad (5)$$

Since $b|a = \{x, a \wedge b \leq x \leq \neg a \vee b\}$ and $d|c = \{y, c \wedge d \leq y \leq \neg c \vee d\}$, the following natural partial (interval-like) ordering as proposed by (Goodman and Nguyen, 1988), can be defined on $\mathcal{B} \setminus \mathcal{B}$ and will be also denoted by " \leq "

$$b|a \leq d|c \Leftrightarrow a \wedge b \leq c \wedge d \text{ and } \neg a \vee b \leq \neg c \vee d \quad (6)$$

This definition extends the partial ordering in \mathcal{B} which is recovered by letting $a=c=T$ (it is why we use the same notation). The relation \leq defined by (6) is reflexive and transitive. This ordering relation can be understood as an entailment relation. Indeed, $a \wedge b \leq c \wedge d$ means that each time $b|a$ is true $d|c$ is true also, and $\neg a \vee b \leq \neg c \vee d \Leftrightarrow c \wedge \neg d \leq a \wedge \neg b$ means that each time $d|c$ is false, $b|a$ is false too. It can be checked that (6) is in agreement with (4), i.e.

$$(b|a \leq d|c \text{ and } d|c \leq b|a) \Leftrightarrow b|a = d|c \quad (7)$$

Besides, in terms of truth-table, the relation \leq defined by (6) corresponds to the ordering $0 \leq ? \leq 1$, still using the same symbol " \leq " between values of the truth function, as pointed out in (Dubois and Prade, 1990a). See the appendix for a formal proof. In particular, we have

$$\forall c, b|a \leq (b \vee c)|a \quad (8)$$

It contrasts with the following fact:

there is no universal ordering between $b|a \wedge c$ and $b|a$ (9)

Indeed $a \wedge b \wedge c \leq a \wedge b$ but $\neg a \vee b \leq \neg a \vee \neg c \vee b$. In other words, $b|a$ may be true as well as false while $b|a \wedge c$ is undefined. This contrasts with the material implication for which $a \rightarrow b$ true entails that $(a \wedge c) \rightarrow b$ is true. The state of fact (9) leaves room for a form of non-monotonicity. But, it can be seen on truth-tables that we cannot have the rules $b|a$ and $\neg b|a \wedge c$ simultaneously true. This is not very surprising since the probabilities $P(b|a)=1$ and $P(\neg b|a \wedge c)=1$ are incompatible in

probability theory ; indeed the former entails $P(\neg b \wedge a) = 0$ and thus $P(\neg b \wedge a \wedge c) = 0$, while the latter entails $P(a \wedge b \wedge c) = 0$; together they lead to $P(a \wedge c) = 0$, making $P(\neg b \wedge a \wedge c)$ undefined. But, a numerical uncertainty measure g compatible with the partial ordering in $\mathcal{B}\mathcal{I}\mathcal{B}$, will remain free, due to (9), to be or not to be such that $g(b \wedge a \wedge c) < g(b \wedge a)$; see Section 4 .

However there is a situation where there exists an ordering between two conditional objects with the same left part. Namely

$$b \wedge c = \perp \Rightarrow b!(a \vee c) \leq b!a \quad (10)$$

(since $b!(a \vee c) = b \wedge a$ and $(\neg a \wedge \neg c) \vee b \leq \neg a \vee b$).

Besides, we have the following ordering which is easy to check

$$(a \wedge b) \mid T \leq b!a \leq (\neg a \vee b) \mid T \quad (11)$$

and enables us to compare $b!a$ with $a \wedge b$ and with $a \rightarrow b$ in $\mathcal{B}\mathcal{I}\mathcal{B}$ in a way that is in agreement with our intuition. Lastly, the following property, which is easy to check, expresses the compatibility of the entailment relation \leq with the union operation performed on the left side of the conditioning symbol "!".

$$\left\{ \begin{array}{l} c!a \leq e!b \\ d!a \leq f!b \end{array} \right. \Rightarrow (c \vee d)!a \leq (e \vee f)!b \quad (12)$$

which extends (8) (recovered for $b=a$, $e=c$, $d=\perp$ in (12)).

2.3 CONJUNCTION OF CONDITIONAL OBJECTS AND BASIC RELATIONS

The problem of extending operations such as negation, conjunction and disjunction to $\mathcal{B}\mathcal{I}\mathcal{B}$ was addressed by Calabrese (1987) and Goodman and Nguyen (1988) among others. There is a consensus about negation, i.e.

$$\neg(b!a) = (\neg b)!a \quad (13)$$

In terms of a truth-table, it corresponds to extending the negation operation by postulating that $t(\neg(b!a)) = ?$ if $t(b!a) = ?$. (13) contrasts with the material implication where $\neg(a \rightarrow b) = a \wedge \neg b \neq a \rightarrow \neg b$. Here, the negation yields a conditional object with the same condition part and an opposite conclusion.

The definition of a conjunction¹ operation for conditional objects is a less straightforward problem. Three possible operations have been proposed by different authors. See Dubois and Prade (1989, 1990a) where it is pointed out that these three proposals correspond to three manners for extending the binary conjunction operation, so as to

¹ The disjunctions are not considered here ; see Dubois and Prade (1990a, 1991)

preserve the symmetry and to take into account the third value "?". Namely, we can define three operations denoted respectively by \wedge , \cdot , \cap

$$1 \wedge ? = 1 \quad ; \quad 0 \wedge ? = 0 \quad ; \quad ? \wedge ? = ? \quad (14)$$

$$1 \cdot ? = ? \quad ; \quad 0 \cdot ? = 0 \quad ; \quad ? \cdot ? = ? \quad (15)$$

$$1 \cap ? = ? \quad ; \quad 0 \cap ? = ? \quad ; \quad ? \cap ? = ? \quad (16)$$

As it can be seen these three operations correspond to the three possible locations of "?" with respect to 0 and 1, in the sense of the conjunction operation. With \wedge , "?" is "above" both 0 and 1, with \cdot , it is in between, while with \cap , "?" is "below" both 0 and 1. Thus only \cdot is in agreement with the ordering defined by \leq in the sense that $t(b!a) \cdot t(d!c) = t(b!a) \Leftrightarrow t(b!a) \leq t(d!c)$. These three conjunction operations are idempotent and associative and were first considered in the framework of trivalent logics by Sobociński, Lukasiewicz and Bochvar respectively. The conjunction of two conditional objects such that at least one of them is undefined, remains undefined in any case with \wedge ; the result with \cdot is undefined only when one object is undefined and the other object is not false ; with \cap the result is undefined only if both objects are undefined. As said in Dubois and Prade (1990a, 1991), the three conjunctions defined above in terms of truth values are equivalently defined in terms of conditional objects (we use the same symbol between conditional objects or between truth values) :

$$(b!a) \wedge (d!c) = ((a \rightarrow b) \wedge (c \rightarrow d)) \mid (a \vee c) \quad (17)$$

$$(b!a) \cdot (d!c) = (b \wedge d) \mid ((a \wedge \neg b) \vee (c \wedge \neg d) \vee (a \wedge c)) \quad (18)$$

$$(b!a) \cap (d!c) = (b \wedge d) \mid (a \wedge c) \quad (19)$$

Indeed it can be checked on truth-tables that $t((a \rightarrow b) \wedge (c \rightarrow d)) \mid (a \vee c) = t(b!a) \wedge t(d!c)$ where \wedge is defined by (14) ; a similar result holds for (18) and (15), and for (19) and (16). Let us observe that $b!a \mid T = (a \wedge b) \mid a = b!a$ and unfortunately this is not equal to $(a \wedge b) \mid T$ as expected. This is not surprising if we identify $a \in \mathcal{B}$ with $a \mid T \in \mathcal{B}\mathcal{I}\mathcal{B}$ and we remember that the operation $*$ in (1) should be such that $? * 0 = 0$, while we have that $? \cap 0 = ?$. Thus the operation \cap is not suitable for our conditional objects.

As pointed out in (Dubois and Prade, 1991), the operations ' \wedge ' and ' \cdot ' are monotonic with respect to \leq , i.e.

$$\left\{ \begin{array}{l} b!a \leq d!c \\ f!e \leq h!g \end{array} \right. \Rightarrow b!a \wedge f!e \leq d!c \wedge h!g \quad (20)$$

the same holds for ' \cdot ' (but would not for ' \cap ' since $0 \cap ? (=?) \leq 0 \cap 1 (=0)$ does not hold). This can be easily checked on truth-tables.

Note that the conjunction defined by (17) (first proposed by Calabrese (1987)) can be put under an equivalent form, indeed it can be shown that

the other hand. However, the counterpart of the reflexivity property is less straightforward.

In the patterns of non-monotonic inference, reflexivity means that terms of the form $a \sim a$ can always be added to a set of non-monotonic deductions $a_i \sim b_i$, and by the right weakening axiom, any term of the form $a \sim (a \vee b)$ can also be added. Conditional objects do not satisfy the reflexivity property, but only a weaker property, since the conditional object $a \sim a$ is undefined if a is false, i.e. $t(a \sim a) \in \{?, 1\}$. With conditional objects the reflexivity axiom indeed comes down to verifying that $(bla) \sim (clc) = bla$. This only holds if $c \rightarrow a = T$ or equivalently $\models c \rightarrow a$ (since $bla \sim clc = (a \rightarrow b) \sim (a \vee c)$ and $(a \rightarrow b) \sim (a \vee c) = (a \rightarrow b)la = bla$ only if $a \vee c = a$); moreover

$$\text{if } \models c \rightarrow a, (bla) \sim (d \vee clc) = bla. \quad (31)$$

also holds more generally. Hence the reflexivity axiom of C must be weakened into an inference rule that specifies when conditional objects of the form clc can be added to a set of premises. This can be achieved through the following inference rule

$$\frac{\models c \rightarrow a ; a \sim b}{c \sim c} \quad (\text{weak reflexivity})$$

which corresponds to the inequality $bla \leq clc \wedge bla$ when $\models c \rightarrow a$. In the above rule we omitted to add $a \sim b$ again below the horizontal line. Indeed the ";" symbol and the horizontal line indicating higher-order entailment in the C patterns are such that it is possible to simultaneously add a term $a \sim b$ above the horizontal line and below it. This is consistent with the fact that operation \sim is monotonic with respect to \leq .

In a non-monotonic logic, the deduction operation \sim is non-monotonic, but the higher level deduction operation using $(;,-)$ has all the properties of a well-behaved classical deduction operation, i.e. letting A be a set of non-monotonic deduction rules $a_i \sim b_i$, we can check that

$$\begin{aligned} A &\text{ (reflexivity)} ; A & \text{ and } B & \text{ imply } C & \text{ (transitivity);} \\ &\text{ and } \frac{A}{B} & \text{ imply } \frac{A \cup C}{B} & \text{ (monotonicity)} \end{aligned}$$

It is clear that \leq does satisfy transitivity and reflexivity properties for conjunctions of conditional objects (since these reduce to conditional objects, and \leq is a reflexive and transitive relation). However, monotonicity of \leq holds only under a restrictive condition, namely we have, as it can be checked

$$\text{if } bla \leq dlc \text{ and } \models e \rightarrow a, \text{ then } bla \sim fle \leq dlc \quad (32)$$

Indeed the condition $\models e \rightarrow a$ blocks the possibility to

simultaneously have $t(bla) = ?$ and $t(fle) = 1$ (since $t(fle) = 1$ requires $t(e) = 1$ and then $t(a) = 1$ and $t(bla) \in \{0, 1\}$); it would cause the violation of the ordering relation in (32) when $t(dlc) = ?,$ because $? \wedge 1 = 1 \geq ?$ (32) corresponds to adding a conditional object that is more specific than the ones already present in the knowledge base. More generally, $bla \leq dlc$ entails $bla \sim fle \leq dlc$ if and only if $\models e \rightarrow (a \vee c)$. Note that (32) holds without the condition $\models e \rightarrow a$ for the conjunctive operation ' $\cdot \cdot$ ' which unfortunately has not a behavior compatible with the idea of non-monotonic reasoning as already said. Clearly (32) generalizes (31).

In the system C, Kraus et al. (1990) have shown that the following rules hold as consequences of the reflexivity axiom and of the four inference rules of the system :

$$\frac{a \sim b ; a \sim c}{a \sim b \wedge c} \quad (\text{"right and"})$$

$$\frac{a \sim b ; b \sim a ; b \sim c}{a \sim c} \quad (\text{"reciprocity", Mackinson (1989)})$$

These two rules have direct analogues in terms of conditional objects, as expressed by (23) and (28) respectively. Moreover, they can be formally derived using the four inference rules of C, and the weak reflexivity rule only. These derivations are made in (Dubois and Prade, 1990a) in the (\wedge, \leq) notation (owing to the idempotence of \wedge and the transitivity of \leq); here we do them in the style of non-monotonic logic .

Derivation of "right hand" (23)

A1) $a \sim b$; A2) $a \sim c$; T1) $a \wedge b \sim a \wedge b \wedge c$ (weak reflexivity) ; T2) $a \wedge b \sim b \wedge c$ (right weakening from T1) ; T3) $a \wedge b \sim c$ (restricted monotonicity from A1 and A2) ; T4) $a \wedge b \sim b \wedge c$ (cut on T2 and T3) ; T5) $a \sim b \wedge c$ (cut on A1 and T4). Q.E.D.

Derivation of "reciprocity" (28)

A1) $a \sim b$; A2) $b \sim a$; A3) $b \sim c$; T1) $b \sim a \wedge c$ (right and on A2, A3) ; T2) $a \wedge b \sim a \wedge c$ (restricted monotonicity on A2 and T1) ; T3) $a \sim a \wedge c$ (cut on A1 and T3) ; T4) $a \sim c$ (right weakening). Q.E.D.

As we can see, the weak form of reflexivity (namely (31)) available in the conditional object framework is sufficient for deriving inference rules expressing (23) and (28).

Moreover, following Kraus et al. (1990) again, (25) relates to the deduction theorem, and corresponds to the rule :

$$\frac{a \wedge b \sim c}{b \sim a \vee c} \quad (\text{HD1})$$

These authors indicate that the converse inference (which

does not hold for conditional objects) leads to the collapse of the C system onto classical logic. Lastly, the analogue of the property (27) of conditional objects for the weak consequence relation, namely

$$\frac{a \sim c ; b \sim c}{a \vee b \sim c} \quad (\text{distribution})$$

has been proved to be a consequence of the (HD1) property added to the C systems by Kraus et al. (1990). Indeed it is possible to derive (27) without the reflexivity axiom, using left equivalence and right weakening and the counterparts of (24), (25), (26), (31).

Derivation of "distribution" (27)

A1) $a \sim c$; A2) $b \sim c$; T1) $a \wedge (a \vee b) \sim c$ (left equivalence on A1) ; T2) $b \wedge (a \vee b) \sim c$ (left equivalence on A2) ; T3) $a \vee b \sim \neg a \vee c$ (HD1 on T1) ; T4) $a \vee b \sim \neg b \vee c$ (HD1 on T2) ; T5) $a \vee b \sim (\neg a \vee c) \wedge (\neg b \vee c)$ ('right and' on T3 and T4) ; T6) $a \vee b \sim a \vee b$ (weak reflexivity on T5) ; T7) $a \vee b \sim (a \vee b) \wedge c$ ('right and' on T5 and T6) ; T8) $a \vee b \sim c$ (right weakening). Q.E.D.

Besides, using \sim in place of ";" the counterparts of the transitivity rule $\frac{a \sim b ; b \sim c}{a \sim c}$, of the contraposition rule

$\frac{a \sim b}{\neg b \sim \neg a}$, and of one half of the deduction theorem, namely

$\frac{b \sim a \rightarrow c}{a \wedge b \sim c}$ do not hold for conditional objects.

This is satisfying because these inference rules cannot be derived from the system C. However the transitivity holds with the conjunction ' \cdot ' as already said. This does not necessarily mean a complete collapse of the calculus of conditional objects based on this conjunction into a fully monotonic calculus, since for instance the contraposition rule and the half above of the deduction theorem, which do not depend on a particular conjunction do not hold for conditional objects ; moreover remind that the reflexivity axiom does not hold for conditional objects and is needed for the collapsing results in C. This is why all the results about collapsing the system C into a monotonic system are not necessarily true for conditional objects with the conjunction ' \cdot '.

Lastly, the negation of conditional objects defined by (13), namely $\neg(b|a) = (\neg b|a)$ has clearly a probabilistic connotation, whose intended meaning is not the counterpart of a rule that would be of the form (where \sim is short for "not~")

$$\frac{a \sim b}{a \sim \neg b}$$

which is never assumed in non-monotonic reasoning. Indeed the negation of conditional objects is such that $\neg 1=0$, while \sim might rather correspond, in terms of conditional objects, to a negation expressing that $b|a$ is false or undefined.

As suggested by the results presented in this section conditional objects seem to offer a promising framework for non-monotonic reasoning. For instance from the "rule" base of conditional objects $(\text{bird}|penguin, \text{fly}|bird, \neg\text{fly}|penguin)$ and the "factual" base that Tweety is a bird and a penguin, we are able to derive (using (24)) that $\neg\text{fly}(\text{bird} \wedge \text{penguin})$. Note that the condition part of the conditional object which is deduced should always correspond to the whole content of the factual base.

4 NUMERICAL SEMANTICS OF NON-MONOTONIC REASONING

A numerical structure for quantifying uncertainty can be put on the top of conditional objects provided that the measure of uncertainty g we use remains in agreement with the entailment relation \leq , i.e. that we have

$$b|a \leq d|c \Rightarrow g(b|a) \leq g(d|c) \quad (33)$$

where $g(\cdot|a)$ and $g(\cdot|c)$ are conditional mappings from \mathcal{B} to the real interval $[0,1]$ which can be defined in terms of a mapping g from \mathcal{B} to $[0,1]$. In this section, we consider the cases where g is a probability measure, a possibility measure, or the dual necessity measure.

4.1 CONDITIONAL OBJECTS AND PROBABILITY THEORY

It can be easily proved that (33) is satisfied by probability measure P from the definition of \leq and writing $P(b|a) = \frac{1}{1+P(\neg a \vee b)}$; see (Dubois and Prade, 1990a).

Probabilistic counterparts of (8), (11), (13), (23), (24), (25), (26) are well-known, they are respectively

$$(8) : \forall c, P(b|a) \leq P(b \vee c|a)$$

$$(11) : P(a \wedge b) \leq P(b|a) = \frac{P(a \wedge b)}{P(a)} \leq P(a \rightarrow b)$$

$$(13) : P(\neg b|a) = 1 - P(b|a)$$

$$(23) : P(b \wedge c|a) \in [\max(0, P(b|a) + P(c|a) - 1), \min(P(b|a), P(c|a))]$$

$$(24) : \max(0, P(b|a) + P(c|a) - 1) \leq P(b \wedge c|a) \leq P(c|a \wedge b)$$

$$(25): P(c|a \wedge b) \leq P(\neg a \vee c|b)$$

$$(26): P(b \wedge c|a) = P(b|a) \cdot P(c|a \wedge b) \quad (\text{Bayes rule})$$

As we can see, conditional objects offer a perfect symbolic version, i.e. (26), of Bayes rule. In spite of this excellent agreement between conditional objects and probabilities, there is not a uniform "translation" of \wedge in the probability calculus (e.g. in the counterpart of (23), we get both $\max(0, \cdot + \cdot - 1)$ and \min ; in (26) the product expresses the conjunction).

Probabilistic counterparts of (27), (28), (29), (30) can also be found :

$$(27): P(c|a) > \varepsilon > 0, P(c|b) > \varepsilon' > 0 \Rightarrow \exists \varepsilon'' > 0, P(c|a \vee b) > \varepsilon''.$$

$$(28): P(b|a) \cdot \max(0, 1 - \frac{1 - P(c|b)}{P(a|b)}) \leq P(c|a)$$

(See Dubois and Prade (1988b) for the proof of this inequality)

$$P(c|a \wedge b) + P(c|a \wedge \neg b) \geq P(c|a)$$

$$(29): \left\{ \begin{array}{l} \forall \varepsilon > 0, |P(c|a \wedge b) - P(c|a \wedge \neg b)| \leq \varepsilon \\ \Rightarrow |2P(c|a) - P(c|a \wedge b) - P(c|a \wedge \neg b)| \leq \varepsilon \end{array} \right.$$

$$(30): P(\neg c|a \wedge b) \geq 1 - \frac{1 - P(\neg b|a \wedge c)}{P(b|a)}$$

However, as already pointed out the probabilistic counterparts of relationships between conditional objects cannot be obtained by a straightforward correspondence. A better agreement is obtained with Adams' ε -semantics.

In Adams (1975)' conditional logic, $b|a$ is interpreted as $P(b|a) \geq 1 - \varepsilon$ where ε is arbitrarily close to 0, and denoted as $a \rightarrow b$. This interpretation is much more demanding than the one underlying conditional objects. However Adams (1975) found inference rules that are exact counterparts of (24), (26) and (27), namely :

triangularity : $a \rightarrow b, a \rightarrow c \Rightarrow (a \wedge b) \rightarrow c$

Bayes rule : $a \rightarrow b, (a \wedge b) \rightarrow c \Rightarrow a \rightarrow c$

disjunction : $a \rightarrow c, b \rightarrow c \Rightarrow (a \vee b) \rightarrow c$

The reflexivity axiom (i.e. $a \rightarrow a$) cannot be added, strictly speaking, since $P(a|a)=1$ only if $a \neq \perp$. These rules are used by Pearl (1988) to build a probabilistic-like default logic. See Geffner (1989) for a detailed study in a non-monotonic reasoning framework. This is not very satisfying in practice from a probabilistic point of view, since it assumes that the probability of encountering an exception is always arbitrarily small. However we shall see in the last subsection that this interpretation, does not owe anything to the additivity of probability calculus.

We may think of other probabilistic interpretations of the conditional object $b|a$. For instance Neufeld (1990) uses $P(b|a) > P(b)$ for expressing that " a favors b ", as a qualitative probability counterpart of a default rule.

However with this interpretation the restricted monotonicity does not hold. Namely

$$\begin{array}{ll} \text{from} & P(b|a) > P(b) \\ \text{and} & P(c|a) > P(c) \\ \text{it does not follow that} & P(c|a \wedge b) > P(c) \end{array}$$

as proved by the following counter-example: $P(a \wedge b \wedge \neg c) = 0.1 = P(a \wedge \neg b \wedge \neg c), P(\neg a \wedge \neg b \wedge \neg c) = 0.8$. Then $P(a \wedge b) = 0.1 = P(a \wedge c), P(a) = 0.2, P(b) = 0.1$ and $P(c) = 0.1$, and we can check that $P(a \wedge b) = 0.1 > P(a) \cdot P(b) = 0.02, P(a \wedge c) = 0.1 > P(a) \cdot P(c) = 0.02$, while $P(a \wedge b \wedge c) = 0 < P(a \wedge b) \cdot P(c) = 0.01$. Similarly the cut rule is violated, i.e.

$$\begin{array}{ll} \text{from} & P(b|a) > P(b) \\ \text{and} & P(c|a \wedge b) > P(c) \\ \text{it does not follow that} & P(c|a) > P(c) \end{array}$$

A counter-example is obtained for $P(a \wedge \neg b \wedge \neg c) = 0 = P(\neg a \wedge b \wedge \neg c) = 0, P(a \wedge b \wedge c) = 2/7$ and the five other probabilities $P(*a \wedge *b \wedge *c) = 1/7$, where * stands for the presence or the absence of \neg . Then $P(a \wedge b) = 3/7, P(a \wedge c) = 2/7, P(a) = 4/7, P(b) = 4/7, P(c) = 4/7$, and $P(a \wedge b) = 3/7 > P(a) \cdot P(b) = 16/49, P(a \wedge b \wedge c) = 2/7 > P(a \wedge b) \cdot P(c) = 12/49$, while $P(a \wedge c) = 2/7 < P(a) \cdot P(c) = 16/49$.

We may think of another probabilistic interpretation of weak statement of the form "if a then b generally", namely $P(b|a) > P(\neg b|a)$; see Wellman (1990) for instance. However, again this interpretation disagrees with minimal requirements of non-monotonic reasoning. For instance, the restricted monotonicity does not hold, i.e.

$$\begin{array}{ll} \text{from} & P(b|a) > P(\neg b|a) \\ \text{and} & P(c|a) > P(\neg c|a) \\ \text{it does not follow that} & P(c|a \wedge b) > P(\neg c|a \vee \neg b) \end{array}$$

Indeed let us take $P(a \wedge b \wedge \neg c) = x = P(a \wedge \neg b \wedge c) > 0$ and $P(\neg a \wedge b \wedge \neg c) = 1 - 2x$. Then $P(a \wedge b)(1 - P(a)) = x(1 - 2x) > P(\neg a \wedge b) \cdot P(a) = 0$ and $P(a \wedge c)(1 - P(a)) = x(1 - 2x) > P(\neg a \wedge c) \cdot P(a) = 0$, while $P(a \wedge b \wedge c)(1 - P(a \wedge b)) = 0 < P(c \wedge \neg(a \wedge b)) \cdot P(a \wedge b) = x^2$.

Yet another interpretation of the default rule "if a then b " is $P(b|a) > P(\neg b|a)$; this is stronger than $P(b|a) > 0.5$ which is obviously not preserved by the inference rules. All these negative results tend to indicate that probabilistic reasoning offers a framework for default reasoning which strongly differs from non-monotonic reasoning according to Gabbay's and Kraus et al.' requirements when probabilities are no longer infinitesimal.

4.2 POSSIBILISTIC SEMANTICS OF NON-MONOTONIC REASONING

Let Π be a function from the Boolean algebra \mathcal{B} to the unit interval such that

- i) $\Pi(\perp)=0 ; \Pi(T)=1$
- ii) $\Pi(a \vee b)=\max(\Pi(a), \Pi(b))$

This function is a measure of uncertainty introduced by Zadeh (1978) under the name possibility measure. The following conventions are generally adopted : $\Pi(a)=0$ means that a is impossible (hence false), while $\Pi(a)=1$ only means that a is possible ; particularly it is allowed to have $\Pi(a)=\Pi(\neg a)=1$, which expresses complete ignorance about a . The quantity $N(a)=1-\Pi(\neg a)$ is called the degree of necessity (or certainty) of a and satisfies an axiom dual to ii), i.e. $N(a \wedge b)=\min(N(a), N(b))$. Note that $\Pi(a \wedge b)$ is not necessarily a function of $\Pi(a)$ and $\Pi(b)$ since a and b can be viewed as possible even when $a \wedge b$ turns out to be impossible. Possibility and necessity measures are special cases of belief functions where consonance is assumed (see Shafer, 1976 ; Dubois and Prade, 1988c), and are characterized by normal fuzzy sets $\mu_F: U \rightarrow [0,1]$, such that \mathcal{B} is the set of subsets A of U , i.e.

$$\Pi(A)=\max\{\mu_F(u) | u \in A\}$$

where $\mu_F(u)=1$ for some $u \in U$ (normalisation), and A can be viewed as the set of models of a .

Putting together possibility measures and logic leads to a logic of incomplete information called possibilistic logic (e.g. Dubois and Prade, 1990b), where incompleteness is described in terms of fuzzy sets of interpretations.

Conditional possibility can be defined similarly to conditional probability from the "Bayes rule" property of conditional objects, i.e. (26) namely

$$\Pi(a \wedge b)=\Pi(b|a) * \Pi(a) \quad (34)$$

where $*$ is a given operation on the unit interval which must be associative, monotonically increasing in the wide sense for both places, and such that $x * 1=x$, $x * 0=0$. If continuity is assumed $*$ must be a triangular norm (Schweizer and Sklar, 1983), and in practice only $*=\text{minimum}$ and $*=\text{product}$ are possible, up to isomorphisms (see Dubois and Prade, 1990a, 1991).

When $*=\text{product}$, $\Pi(b|a)$ corresponds to $\Pi(a \wedge b)/\Pi(a)$, i.e. Dempster rule of conditioning (Shafer, 1976). When $*=\text{min}$, it corresponds to the definition of Hisdal (1978) and $\Pi(b|a)$ is not uniquely determined. However, in (Dubois and Prade, 1988c), $\Pi(b|a)$ is defined as the least specific solution of (34) with $*=\text{min}$, i.e.

$$\begin{aligned} \Pi(b|a) &= 1 \text{ if } \Pi(a \wedge b)=\Pi(a) \\ &= \Pi(a \wedge b) \text{ if } \Pi(a \wedge b)<\Pi(a) \end{aligned} \quad (35)$$

This definition tries to allocate the greatest degree of possibility to event b in the context a . Conditional necessity is defined by duality as $N(b|a)=1-\Pi(\neg b|a)$ in

whether $*=\text{min}$ or product.

The two definitions of conditioning completely agree when $\Pi(b|a)=1$ since with $*=\text{product}$ we still have $\Pi(b|a)=1$ if and only if $\Pi(a \wedge b)=\Pi(a)$. In the following, results will apply with both definitions of conditioning, unless otherwise specified. A default rule "if a then b up to exceptions" will here be translated into

$$N(b|a)>0.$$

Due to the properties of necessity measures it implies that $N(\neg b|a)=0$, i.e. $\Pi(\neg b|a)=1>\Pi(\neg b|\neg a)$; this is the counterpart of (13). Hence this translation means that given a context where a is true, b is more certain than $\neg b$, or equivalently $\neg b$ is less possible than b . This view of default rules sounds much weaker and much more natural than the one suggested by Adams' ε -semantics.

Let us check that this definition is in accordance with conditional objects, and leads to a non-probabilistic model of the system C+(HD1)

$$\bullet b|a \leq d|c \Rightarrow \Pi(b|a) \leq \Pi(d|c) \quad (36)$$

Proof : for $*=\text{product}$ see Dubois and Prade (1990a). When $*=\text{min}$, we only have to make sure that $\Pi(b|a)=1 \Rightarrow \Pi(d|c)=1$. Let us note that $\Pi(a)=\max(\Pi(a \wedge b), \Pi(a \wedge \neg b))$. Assume $\Pi(a \wedge b)=\Pi(a)$; it follows that $\Pi(a \wedge b) \geq \Pi(a \wedge \neg b)$. If $b|a \leq d|c$, the monotonicity of Π and the transitivity leads to $\Pi(c \wedge d) \geq \Pi(c \wedge \neg d)$ since $a \wedge b \leq c \wedge d$ and $a \wedge \neg b \geq c \wedge \neg d$. Then we have

$$\Pi(c)=\max(\Pi(c \wedge d), \Pi(c \wedge \neg d))=\Pi(c \wedge d), \text{ i.e. } \Pi(d|c)=1 \quad \text{Q.E.D.}$$

$$\bullet b|a \leq d|c \Rightarrow N(b|a) \leq N(d|c) \quad (\text{obvious by duality})$$

A systematic translation of the inference rules of C can be checked as follows

$$\bullet \text{restricted monotonicity} :$$

$$N(b|a)>0 \text{ and } N(c|a)>0 \text{ imply } N(c|a \wedge b)>0 \quad (37)$$

Proof : the premises are equivalent to $\Pi(a)>\Pi(a \wedge \neg b)$ and $\Pi(a)>\Pi(a \wedge \neg c)$. Hence $\Pi(a)=\max(\Pi(a \wedge b), \Pi(a \wedge \neg b))=\Pi(a \wedge b)$. Finally $\Pi(a \wedge b)>\Pi(a \wedge \neg c) \geq \Pi(a \wedge b \wedge \neg c)$, which leads to $\Pi(\neg c|a \wedge b)<1$. Q.E.D.

$$\bullet \text{cut}:$$

$$N(b|a)>0 \text{ and } N(c|a \wedge b)>0 \text{ imply } N(c|a)>0 \quad (38)$$

Proof (Dubois and Prade, 1990d) : the premises are equivalent to $\Pi(a)>\Pi(a \wedge \neg b)$ and $\Pi(a \wedge b)>\Pi(a \wedge b \wedge \neg c)$. Hence $\Pi(a \wedge \neg c)=\max(\Pi(a \wedge \neg c \wedge b), \Pi(a \wedge \neg c \wedge \neg b)) \leq \max(\Pi(a \wedge b \wedge \neg c), \Pi(a \wedge \neg b))<\max(\Pi(a \wedge b), \Pi(a))=\Pi(a)$. Q.E.D.

The left equivalence and right weakening rules trivially

hold. Moreover the weak reflexivity inference rule cannot be strengthened into a reflexivity axiom, strictly speaking, since $N(a|a)=1$ if $a \neq \perp$ and $N(a|a)=0$ if $a=\perp$. Lastly as a consequence, the distribution rule (or disjunction rule) works with conditional possibility, i.e.

• distribution :

$$N(c|a)>0 \text{ and } N(c|b)>0 \text{ imply } N(c|a \vee b)>0 \quad (39)$$

Direct proof : $\Pi(a)>\Pi(\neg c \wedge a)$ and $\Pi(b)>\Pi(\neg c \wedge b)$ imply

$$\begin{aligned} \Pi(a \vee b) &= \max(\Pi(a), \Pi(b)) > \max(\Pi(\neg c \wedge a), \Pi(\neg c \wedge b)) = \\ &= \Pi(\neg c \wedge (a \vee b)). \end{aligned}$$

Q.E.D.

The above results can be stated in a much stronger way by means of inequalities when conditional possibility is defined by means of the minimum operation, i.e., (35).

• restricted monotonicity :

If $N(b|a)>0$ and $N(c|a)>0$ then

$$N(c|a \wedge b) \geq \min(N(b|a), N(c|a)) \quad (40)$$

Proof: It is enough to use the previous proof where the following inequalities have been established:

$$\Pi(a \wedge b) > \Pi(a \wedge \neg c) \geq \Pi(a \wedge b \wedge \neg c),$$

from which two things follow: $\Pi(a \wedge b \wedge \neg c) \leq \max(\Pi(a \wedge \neg c), \Pi(a \wedge b))$ and $\Pi(a \wedge b \wedge \neg c) = \Pi(\neg c|a \wedge b)$. Using the assumptions, $\Pi(\neg c|a) = \Pi(a \wedge \neg b)$, $\Pi(\neg c|b) = \Pi(a \wedge \neg c)$. So, (40) obtains by duality. Note that we have actually proved $N(c|a \wedge b) \geq N(c|a)$ if $N(b|a)>0$; this does not express monotonicity due to the presence of the condition $N(b|a)>0$.

Q.E.D.

• cut

If $N(b|a)>0$ and $N(c|a \wedge b)>0$ then

$$N(c|a) \geq \min(N(b|a), N(c|a \wedge b)) \quad (41)$$

Proof: It is again enough to use the previous proof where the following inequalities have been established:

$$\Pi(a \wedge \neg c) \leq \max(\Pi(a \wedge b \wedge \neg c), \Pi(a \wedge \neg b)) < \Pi(a)$$

Hence $\Pi(\neg c|a) = \Pi(a \wedge \neg c)$. Using the assumptions and duality again, (41) follows.

Q.E.D.

• distribution :

$N(c|a)>0$ and $N(c|b)>0$ imply

$$N(c|a \vee b) \geq \min(N(c|a), N(c|b)) \quad (42)$$

Proof: Obvious due to the definition of necessity measures. In fact the inequality is an equality since $N(c|a) = N(\neg a \vee c)$ when positive and $(\neg a \vee c) \wedge (\neg b \vee c) = \neg(a \vee b) \vee c$.

Q.E.D.

Note that (40-42) obviously still hold if the strict inequalities " >0 " are changed into inequalities " ≥ 0 ". This result suggests a systematic translation of (\wedge, \leq) in the unit interval, namely $b|a$ is interpreted as $N(b|a)>0$, \wedge corresponds to min, and \leq corresponds to the inequality in the reals. It also suggests a systematic way of translating inference between non-monotonic deductions such as

$$\frac{a \vdash b \ ; \ c \vdash d}{e \vdash f}$$

into $N(f|e) \geq \min(N(b|a), N(d|c))$. This state of facts completely contrasts with the probabilistic semantics of non-monotonic inference, where no systematic translation other than with infinitesimal probabilities has been found yet. However (40-42) generally do not hold with Dempster rule of conditioning used for the definition of conditional possibility.

What we have proved is that possibility theory offers a setting for interpreting non-monotonic logic which is as natural as probability theory through ε -semantics. Using the definition of conditional possibility based on the minimum operation, the agreement is even deeper. This is not surprising because possibility theory is closely related to epistemic entrenchment (Dubois and Prade, 1990c), and also to Shoham's preference logic (Dubois and Prade, 1990d), and because belief revision axioms, which heavily rely on epistemic entrenchment, are another way of expressing the axioms of non-monotonicity (Gärdenfors, 1990).

4.3 A POSSIBILISTIC INTERPRETATION OF ADAMS' ε -SEMANTICS

In this last section we make one step further, claiming that not only possibility theory offers a natural setting for a model of non-monotonic logic, but that Adams' ε -semantics is just another way of presenting the possibilistic model presented in Section 5. Namely we prove the following result : there is a transformation from possibility to probability such that $N(b|a)>0$ if and only if $P(b|a)>1-\varepsilon$ where ε is positive but infinitely close to 0. To see it, we must use an intermediary step, which is Spohn (1990)'s theory of conditional functions. We consider a special case of Spohn's model under the form of functions $\kappa = \mathbb{B} \rightarrow \mathbb{N}$ (the set of positive integers) such that

$$\begin{aligned} \kappa(a \vee b) &= \min(\kappa(a), \kappa(b)) \\ \kappa(T) &= 0 \end{aligned}$$

$\kappa(a)$ is viewed as an integer-valued degree of impossibility of a . In (Dubois and Prade, 1990c), we indicated that the function $\Pi_K(a) = k^{-\kappa(a)}$, where $k>1$ is

nothing but a possibility measure. On the other hand Spohn (1990) points out that $\kappa(a)=n$ can be viewed as a non-standard degree of probability $P(a)$ such that $P(a) \cdot \epsilon^{-n}$ is finite, where ϵ is infinitesimal. This is denoted $P(a) \sim \epsilon^n$ here. Then it is obvious that if $a \wedge b = \perp$, $P(a) \sim \epsilon^n$ and $P(b) \sim \epsilon^m$ then $P(a \vee b) \sim \epsilon^{\min(n,m)}$, and that this is true even if $a \wedge b \neq \perp$ since $P(a \wedge b) \leq \min(P(a), P(b))$. We recognize the axioms of the κ -functions. Clearly, the properties of infinitesimal probabilities owe little to the additivity axiom itself. $\kappa(a)$ can thus be viewed as the order of magnitude of a very small probability. Putting all the above results together, a degree of possibility strictly between 0 and 1 can thus be interpreted as an infinitesimal probability, by turning it first into a κ -function.

These transformations carry over to conditioning. Indeed conditioning a κ -function by $a \in \mathcal{B}$ leads to define $\kappa(b|a) = \kappa(a \wedge b) - \kappa(a)$ (Spohn, 1990). Using $\Pi_K(a) = k^{-\kappa(a)}$ leads to $\Pi_K(b|a) = \frac{\Pi_K(a \wedge b)}{\Pi_K(a)}$; moreover conditioning a κ -function is in accordance with probability theory since if $P(a \wedge b) = \epsilon^n$, $P(a) = \epsilon^m$ then $P(b|a) = \epsilon^{n-m}$.

Based on these results, it is easy to see that

$$\begin{aligned} & P(b|a) > 1 - \epsilon \text{ for an infinitesimal } \epsilon \\ \Leftrightarrow & P(\neg b|a) \sim \epsilon^n \text{ for } n \geq 1 \\ \Leftrightarrow & \kappa(\neg b|a) = n \text{ for some } \kappa\text{-function} \\ \Leftrightarrow & \Pi(\neg b|a) = k^{-n} < 1 \text{ for some } k > 1 \\ \Leftrightarrow & N(b|a) > 0. \end{aligned}$$

Hence Adams' ϵ -semantics entirely agrees with the structural properties of possibility theory, i.e. owes little to the additive character of probability theory. It is in fact equivalent to a theory of all-or-nothing possibility ($N(b|a) > 0$ or $= 0$). This result, along with the difficulty to interpret the axioms of non-monotonic logic in terms of non-infinitesimal probability values, considerably weakens Pearl (1988)'s hint that probability theory offers natural semantics for non-monotonic reasoning, at least in terms of the axioms proposed by Kraus et al. (1990). Besides the calculus of orders of magnitude of infinitesimal probabilities follows the laws of possibility theory, and Dempster rule for conditioning.

5 CONCLUSION

Conditional objects, which allow a measure-free view of the idea of conditioning, still remaining in agreement with numerical calculi of uncertainty such as probability and possibility theories, are also in agreement with requirements recently settled for non-monotonic inference. Thus conditional objects seem to offer a promising framework for non-monotonic reasoning. Another important result of this paper is that a form of possibility

theory offers a natural semantics for non-monotonic reasoning that turns out to be equivalent to Adams' ϵ -semantics. The advantages of the language of possibility theory is to highlight the ordinal nature of the corresponding non-monotonic logic, without misleadingly suggesting, as in Pearl (1988), that the additivity axiom of probability theory is at work in the inference rules. Lastly, we have shown the existence of a systematic translation of non-monotonic inference rules in the quantitative finistic framework of possibility theory, relying on a special notion of conditioning based on the minimum operation, something that, to-date, does not seem to exist for probability theory.

Appendix

Agreement of definition (6) of entailment with the ordering $0 \leq ? \leq 1$

$$\begin{aligned} b|a \leq d|c & \Leftrightarrow a \wedge b \leq c \wedge d \text{ and } \neg a \vee b \leq \neg c \vee d \\ & \Leftrightarrow t(a \wedge b) \leq t(c \wedge d) \text{ and } t(\neg a \vee b) \leq t(\neg c \vee d) \\ & \Leftrightarrow \text{not}(t(a)=t(b)=1 \text{ and } t(c)=0 \text{ or } t(d)=0) \text{ and} \\ & \quad \text{not}(t(a)=0 \text{ or } t(b)=1 \text{ and } t(c)=1 \text{ and } t(d)=0) \\ & \Leftrightarrow (t(a)=0 \text{ or } t(b)=0 \text{ or } t(c)=t(d)=1) \text{ and} \\ & \quad ((t(a)=1 \text{ and } t(b)=0) \text{ or } t(c)=0 \text{ or } t(d)=1) \\ & \Leftrightarrow (t(a)=0 \text{ or } t(b)=0) \text{ and } (t(c)=0 \text{ or } t(d)=1) \\ & \quad \text{or } t(b)=0 \text{ and } t(a)=1 \\ & \quad \text{or } t(c)=1 \text{ and } t(d)=1 \\ & \Leftrightarrow t(b|a) \in \{0, ?\} \text{ and } t(d|c) \in \{?, 1\} \\ & \quad \text{or } t(b|a)=0 \text{ and } t(d|c) \in \{0, ?, 1\} \\ & \quad \text{or } t(b|a) \in \{0, ?, 1\} \text{ and } t(d|c)=1 \\ & \Leftrightarrow t(b|a) \leq t(d|c) \text{ where } \leq \text{ is defined by the} \\ & \quad \text{ordering } 0 \leq ? \leq 1 \end{aligned} \quad \text{Q.E.D.}$$

Acknowledgements

This work was partially supported by the European ESPRIT Basic Research Action 3085 "DRUMS".

References

- E.W. Adams (1975) *The Logic of Conditionals*. Dordrecht : D. Reidel.
- P. Calabrese (1987) An algebraic synthesis of the foundations of logic and probability. *Information Sciences* 42: 187-237.
- P. Cheeseman (1988) An inquiry into computer understanding. *Computational Intelligence* 4: 58-66.
- D. Dubois & H. Prade (1988a) Conditioning in possibility and evidence theories -A logical viewpoint-. In B. Bouchon, L. Saitta & R.R. Yager (eds.), *Uncertainty and Intelligent Systems*, 401-408. Berlin : Springer Verlag.

- D. Dubois & H. Prade (1988b) On fuzzy syllogisms. *Computational Intelligence* 4(2):171-179.
- D. Dubois H. Prade (1988c) *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. New York: Plenum Press.
- D. Dubois & H. Prade (1989) Measure-free conditioning, probability and non-monotonic reasoning. *Proc. of the 11th Inter. Joint Conf. on Artificial Intelligence*, Detroit, 1110-1114.
- D. Dubois & H. Prade (1990a) The logical view of conditioning and its applications to possibility and evidence theories. *Int. J. of Approximate Reasoning* 4:23-46.
- D. Dubois & H. Prade (1990b) Resolution principles in possibilistic logic. *Int. J. of Approximate Reasoning* , 4: 1-21
- D. Dubois & H. Prade (1990c) Epistemic entrenchment and possibilistic logic. In Report IRIT/90-2/R, IRIT, Univ. P. Sabatier, Toulouse, France. To appear in *Artificial Intelligence*.
- D. Dubois & H. Prade (1990d) Possibilistic logic, preference models, non-monotonicity and related issues. Report IRIT/90-54/R, IRIT, Univ. P. Sabatier, Toulouse.
- D. Dubois & H. Prade (1991) Conditioning, non-monotonic logic and non-standard uncertainty models. In I.R. Goodman, M.M. Gupta, H.T. Nguyen & G.S. Rogers, (eds.), *Conditional Logic in Expert Systems*, to appear. Amsterdam, The Netherlands : North-Holland.
- D.M. Gabbay (1985) Theoretical foundations for non-monotonic reasoning in expert systems. In K.R. Apt (ed.), *Logics and Models of Concurrent Systems*, 439-457. Berlin : Springer Verlag.
- P. Gärdenfors (1990) Belief revision and nonmonotonic logic : two sides of the same coin ?. *Proc. of the 9th Europ. Conf. on Artificial Intelligence (ECAI 90)*, Stockholm, Sweden, August 6-10, 768-773.
- H. Geffner (1989) Default reasoning : causal and conditional theories. Tech. Report 137, Cognitive Systems Laboratory, Department of Computer Science, UCLA, Los Angeles, Ca.
- I.R. Goodman & H.T. Nguyen (1988) Conditional objects and the modeling of uncertainties. In M.M. Gupta & T. Yamakawa (eds.), *Fuzzy Computing - Theory, Hardware and Applications*, 119-138. Amsterdam, The Netherlands : North-Holland.
- I.R. Goodman, H.T. Nguyen & E.A. Walker (1991) *Conditional Inference and Logic for Intelligent Systems* :
- A Theory of Measure-Free Conditioning, to appear. Amsterdam, The Netherlands : North-Holland.
- E. Hisdal (1978) Conditional possibilities: independence and non-interaction. *Fuzzy Sets and Systems*, 1, 283-297.
- K. Kraus, D. Lehmann & M. Magidor (1990) Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44:167-207.
- H. Kyburg (1974) *The Logical Foundations of Statistical Inference*. Dordrecht : D. Reidel.
- Léa Sombé (1990) *Reasoning under Incomplete Information in Artificial Intelligence. A Comparison of Formalisms using a Single Example*. New York:Wiley.
- D. Makinson (1989) General theory of cumulative inference. In M. Reinfranck, J. De Kleer, M.L. Ginsberg & E. Sandewall (eds.), *Non-Monotonic Reasoning* (Proc. of the 2nd Inter. Workshop, Grassau, FRG, June 13-15, 1988), 1-18. Berlin : Springer Verlag.
- E. Neufeld (1990) A probabilistic commonsense reasoner. *Int. J. of Intelligent Systems* 5:565-594.
- J. Pearl (1988) *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. San Mateo, Ca. : Morgan Kaufmann.
- R. Reiter & G. Crisculo (1981) On interacting defaults. *Proc. of the 7th Inter. Joint Conf. on Artificial Intelligence (IJCAI-81)*, Vancouver, 270-276.
- G. Schay (1968) An algebra of conditional events. *J. Math. Anal. & Appl.* 24: 334-344
- B. Schweizer & A. Sklar (1983) *Probabilistic Metric Spaces*. Amsterdam, The Netherlands: North-Holland.
- G. Shafer (1976) *A Mathematical Theory of Evidence*. Princeton University Press, N.J..
- W. Spohn (1990) A general non-probabilistic theory of inductive reasoning. In R.D. Shachter, T.S. Levitt, L.N. Kanal & J.F. Lemmer (eds.), *Uncertainty in Artificial Intelligence 4*, 149-158. Amsterdam, The Netherlands: North-Holland.
- D.S. Touretzky (1984) Implicit ordering of defaults in inheritance systems. *Proc. of the National Conf. on Artificial Intelligence (AAAI-84)*, Austin, Tx., Aug. 6-10, 322-325.
- M.P. Wellman (1990) Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence* 44: 257-303.
- L.A. Zadeh (1978) Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems* 1:3-28.

Completable Representations of Constraint Satisfaction Problems

Eugene C. Freuder
Computer Science Department
University of New Hampshire
Durham, NH 03824

Abstract

A solution to a constraint satisfaction problem specifies values for a set of variables that satisfy constraints on which combinations of values are permitted. Completeness properties specify conditions under which partial solutions, for subsets of variables, will be extensible to full solutions. Different representations for a constraint satisfaction problem can make explicit different sets of constraints. In certain circumstances representations with desirable completeness properties can be obtained in polynomial time.

1 INTRODUCTION

1.1 OVERVIEW

Constraints on a set of variables are relations that specify which combinations of values are permitted for those variables. A *constraint satisfaction problem* (CSP) involves finding an instantiation for each variable such that all the constraints are simultaneously satisfied. Many forms of reasoning can be viewed in CSP terms, e.g. temporal reasoning [Dechter, Meiri and Pearl 89], qualitative reasoning [Kuipers and Berleant 88], truth maintenance [de Kleer 89], diagnostic reasoning [Dechter and Pearl 88b] and image interpretation [Reiter and Mackworth 88].

An instantiation of a subset of variables will be called a *solution* if it satisfies all constraints in which any of the variables appear. A solution for all the variables will be called a *complete solution*. If a solution for a subset can be extended to a complete solution I will call it *completable*. For example if a for X, b for Y and c for Z is a complete solution, then a for X and c for Z is a completable solution for the X,Z subset.

Different representations of a CSP can make explicit different sets of constraints, while remaining *equivalent* in the sense that the set of complete solutions remains the same. These representations can have different completeness properties based on the extent to which

subset solutions are completable. Completeness properties can be desirable, especially when these representations are viewed as knowledge bases subject to multiple queries.

I establish here methods for obtaining CSP representations with various completeness properties. For appropriate classes of CSPs these methods will operate in polynomial time. As an example of the kind of result this work supports: Suppose we are given a scheduling problem involving a sequence of n event variables, each with 7 value choices, in which each event can only constrain the succeeding 5 events. We can obtain in $O(n^3)$ time a representation in which a solution for any subset of more than 5 consecutive events is part of a complete solution.

The notion of completeness, and the methods for establishing it, derive from Montanari's concept of *minimality* [Montanari 74]. A CSP is often represented as a *constraint network*, where vertices correspond to variables and edges to constraints. A network of binary constraints is minimal if each pair of values which satisfies the binary constraint on a pair of variables is completable. Montanari viewed the problem of finding a minimal network equivalent to a given network as the *central problem* for networks of constraints. More recently a concept of minimality has played a role in temporal reasoning [van Beek 89]. Completeness also serves to generalize global consistency [Dechter 90] and (i,n-i)-consistency [Freuder 85].

Though achieving minimality is in general an NP-complete problem, Montanari established certain conditions under which it can be achieved efficiently. He considered "syntactic" restrictions based on the topology or connectivity structure of the constraints in the networks and "semantic" restrictions on the nature or internal structure of the constraints. Dechter has recently shown how semantic restrictions on the number of potential variable values (or the size of the unary constraints) can be related to a generalization of minimality that she terms global consistency [Dechter 90].

In this paper I consider syntactic restrictions. I begin with CSPs whose constraint networks have the structure of an important class of graphs known as "k-trees" [Freuder 90].

K-trees generalize tree structures (trees are 1-trees). Completeness provides appropriate generalizations of minimality in the context of k-trees and I prove completeness results for k-tree-structured CSPs. This leads to a complexity bound on obtaining a completable representation for *any* CSP, expressed as a function of the "k-tree embedability" of its constraint network.

K-tree completeness may prove especially useful in situations where expressivity can be traded for efficiency. Expressivity/efficiency tradeoffs have been studied in the temporal reasoning domain [Vilain, Kautz and van Beek 89]. Tambe and Rosenbloom have utilized results concerning tree-structured CSPs to trade expressivity for efficiency in a production system pattern matcher [Tambe and Rosebloom 90].

For simplicity I assume that the original CSP problems we are given are binary CSPs (non-binary CSPs can be expressed as binary ones [Rossi, Dhar and Petri 89]). Higher order, non-binary constraints will in fact be introduced in transforming to completable representations.

The rest of the paper is organized as follows: Section 1.2 provides some CSP background material and presents a simple, informal example of the kind of result we are seeking, and the types of representation transformation that will be used. Section 2 presents the different representations we use and the transformations employed to go from one to another. It includes further motivation for our interest in k-tree structures. Section 3 introduces the concept of i multi-clique completeness and proves the central theorem of the paper, which shows how to obtain an i multi-clique completable representation of a k-tree in polynomial time (for fixed k). Section 4 examines special cases of multi-clique completeness. It provides a bound for full completeness, where a solution for any subset of variables is completable. Section 5 goes beyond k-trees. It discusses "partial k-trees" and uses them to derive results applicable to any CSP. It discusses series-parallel graphs and bandwidth-k graphs, which may be of special interest in CSP applications, e.g. in temporal reasoning. Section 6 contains a final example and a brief summary.

1.2 BACKGROUND AND EXAMPLE

This section presents a simple example to introduce illustrate the idea of completeness and the nature of the results we will obtain. A "variable completable" representation of a simple CSP is found. At the same time some basic CSP concepts are reviewed.

Consider the graph coloring problem in Figure 1. The problem is to color the vertices of the graph such that no two vertices that are joined by an edge have the same color. At each vertex we can select from the colors indicated. Blue is indicated by "b", red by "r" and green by "g". The vertices of the graph correspond to CSP variables, the edges to CSP constraints.

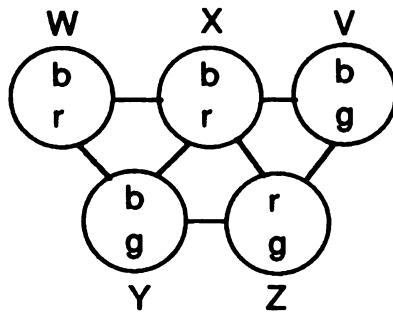


Figure 1: A Graph Coloring Example

It happens that for graph coloring CSPs the constraint network has the same structure as the graph to be colored. This is convenient for devising sample problems with a desired structure.

We will call the constraint network constructed directly from the specification of a CSP the *direct constraint network*. This is to distinguish it from constraint networks we might construct that in some fashion represent the same set of solutions, but with vertices and edges that correspond to a different set of variables and/or constraints. For example, we can modify the direct constraint network by making explicit constraints which are only implicit in the original problem specification, or by removing constraints which are redundant in the original problem specification. (I am trying to avoid in this paper introducing a lot of formal notation and technical detail to specify a CSP and its alternative representations.)

It would be nice to know that any value we pick for any variable that satisfies the constraints directly involving that variable is part of some solution to the problem. We will say that a CSP representation is *variable completable* iff any solution for the singleton subset containing any one variable is completable. This is clearly a nice basic property for a constraint data base to have. However, this is not the case at present. For example, value blue for variable W does not participate in any solution.

To fix this we add further constraints. These are pictured in Figure 2. First, we add ternary (three variable) constraints on the triples of variables WXY, XYZ, XZV. These are the triples of variables that have the property that each variable is constrained by each other variable. The ternary constraints correspond to solutions of three CSP subproblems: each subproblem consists of one of these triples of variables, along with the associated binary constraints between pairs of variables. For example, the ternary constraint on WXY specifies that the triple (red blue green) is a consistent solution to that subproblem.

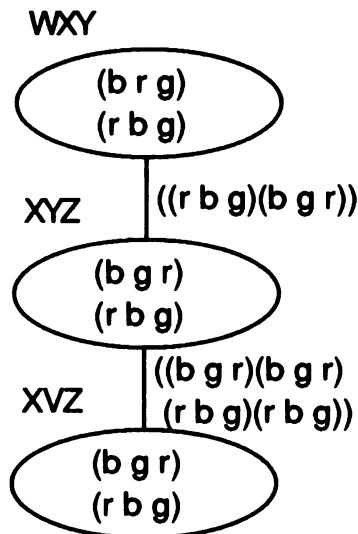


Figure 2: Metaproblem

Next we add a constraint between the subproblems WXY and XYZ, and another between XYZ and XVZ. These new constraints are effectively 4-ary constraints, between 4 variables of the original CSP, as each pair of subproblems involve four distinct variables. However, for the moment we shall view these new constraints as binary constraints in a "metaproblem". The variables of this metaproblem will correspond to the subproblems identified earlier. The values will be the solutions to the subproblems. The constraints will specify that the solutions to the subproblems must agree on common variables. For example, the triple (red blue green) for WXY is consistent with the triple (blue green red) for XYZ, but (blue red green) for WXY is not consistent with any solution for XYZ: it agrees with one on X and the other on Y, but with neither on both X and Y at once.

In Figure 3 we see the result of achieving *arc consistency* [Mackworth 77], or *2-consistency* [Freuder 78], on the metaproblem: any value for a metavariable not consistent with any value at some neighboring metavariable is eliminated. Notice that initially (red blue green) for XVZ is consistent with (red blue green) for XYZ. However, after the constraint between WXY and XYZ eliminates (red blue green) for XYZ, there is nothing left for the (red blue green) solution of XVZ to go with, and it too is eliminated.

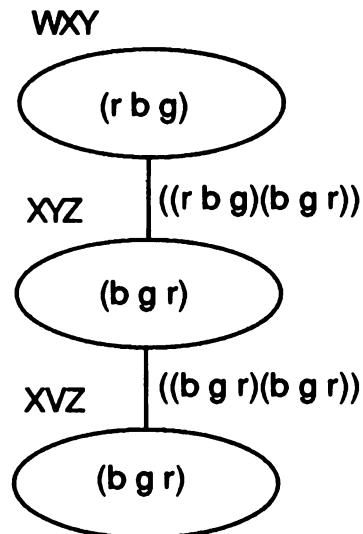


Figure 3: Arc Consistency

I claim that it is now impossible to pick a value for one of the original CSP variables that satisfies the constraints embodied in Figure 3, but fails to be part of a complete solution. Blue for W, for example, is not in any solution for WXY, i.e. does not satisfy the WXY ternary constraint. We could "project" the constraints downward onto the individual variables, e.g. removing blue from W. We can imagine a final representation which contains the original variables, with values removed as called for by the new constraints, along with the new ternary and 4-ary constraints.

At this point you can verify that the representation in Figure 3, is in fact variable completable. However, the process of acquiring this representation remains black magic. Why did I choose those particular subproblems? Why did I link them in that way? How did I know to perform arc consistency processing? These questions will be answered in the rest of the paper. The structure in Figure 3 is an example of what we will define below as a 2-closure of a clique tree representation of the original problem. The original problem has a 2-tree structure. It may help to keep this example in mind as we extend the above process and formulate it in a considerably more general manner to achieve "multi-clique completness".

2 REPRESENTATIONS AND TRANSFORMATIONS

This section defines k-trees, clique trees and i-closures and the representation transformations that take a k-tree into a clique tree and a clique tree into an i-closure of a clique tree. I argue that these transformations do not change the set of solutions. Most of the completeness results will concern closures of clique trees of k-trees. I will use a simple example for illustration.

2.1 K-TREES

This section defines k-trees and discusses their importance. I argue that interesting CSPs can have k-tree structure and that to some degree k-tree results apply to *any* CSP.

Call a graph in which any two vertices are joined by an edge a *clique*. A k-tree is formed by starting with a *k-clique*, a clique of k vertices. Vertices are added by linking them to all the vertices of an already present k-clique. Figure 4 illustrates the construction of a 2-tree. The numbering of the vertices reflects the order in which the 2-tree was built.

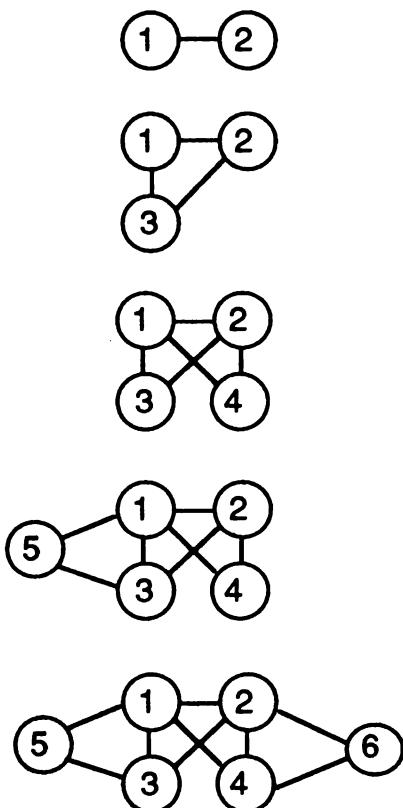


Figure 4: Building a 2-Tree

A graph is a *k-tree* iff:

1. It is a k-clique
or
2. There is a vertex v such that:
 - a. v is connected to a k-clique
 - b. removing v and its associated edges leaves a k-tree.

A *partial k-tree* is a subgraph of a k-tree. A CSP will be called a *k-tree CSP* if the direct constraint network is a k-tree.

Why should we care about k-tree CSPs?

1. 1-trees are the usual tree structures, which can represent hierarchically structured CSPs.

2. 2-trees are series-parallel networks, which might represent e.g. design problems for series-parallel circuits.

3. K-tree CSPs are solvable in time polynomial in k [Freuder 90].

4. Any constraint network can be embedded in a k-tree for large enough k, by adding appropriate trivial constraints (*trivial constraints* allow all value pairs and are not normally represented by constraint network edges; in this case we make an exception).

5. Any constraint network can be reduced to a partial k-tree for any k by removing enough constraints, settling for a partial solution [Freuder 89].

6. Several techniques have been developed to take advantage of tree substructures in general CSPs; these may be profitably extensible to k-trees [Freuder 90]. In particular k-trees provide target structures for decomposing CSPs into subproblems.

7. K-trees structures offer a hierarchy of choices in trading representational power for computational efficiency; tree structures have already been used for this purpose. In some circumstances we may be in a position to guarantee that problems will be k-tree-structured, in return for limiting the class of problems which we can handle.

8. K-trees, not only trees, have a natural "semantics", that makes it likely such structures will arise in practice. For example, if we have a scheduling, planning or temporal reasoning problem involving a sequence of events, with the plausible restriction that one event cannot affect another more than k events distant, we have a partial k-tree. We can make this a k-tree by simply adding trivial constraints as necessary so that each event is linked to all k events on either side.

2.2 CLIQUE TREES

Given a k-tree we can form a *clique tree* representation. The vertices of the clique tree correspond to the maximal cliques ($k+1$ -cliques) in the k-tree. They are linked in a manner which reflects the construction of the k-tree. Adjacent cliques have a k-clique in common. Figure 5 contains a clique tree for the 2-tree in Figure 4. The 1,2,4 node is a child of the 1,2,3 node reflecting the fact that in building the 2-tree we joined the 4 vertex to vertices 1 and 2. (The 2-tree could have been constructed in a different order and has alternative clique tree representations.) A clique tree can be obtained in $O(n)$ time, for an n vertex k-tree; Wimer [Wimer 87] gives a clique tree construction algorithm. The *leaves* are those vertices with a single adjacent edge.

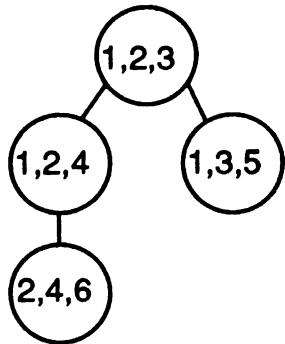


Figure 5: A Clique Tree

This representation is related to acyclic databases and acyclic CSPs [Dechter and Pearl 89], and to tree decompositions [Robertson and Seymour 86]. Its roots go back at least to Beineke and Pippert's early work on k-trees [Beineke and Pippert 71].

The clique tree of a constraint network can be regarded as the constraint network of a *metaproblem* in which the cliques correspond to metavariables whose values are the solutions of the clique subproblems. The constraints of the metaproblem require that consistent values for two adjacent metavariables have identical values for identical variables from the original problem. The clique tree can also be regarded as an expanded version of the original problem, a constraint network in which certain implicit higher level constraints are made explicit; the clique tree vertices correspond to $k+1$ -ary constraints, the clique tree edges to $k+2$ -ary constraints.

2.3 CLOSURES

The *i*-closure of a constraint network results from adding the additional constraints induced by a strong *i*-consistency algorithm [Freuder 78; Freuder 82; Cooper 89]. If a constraint network is strongly *i*-consistent then given any consistent values for a subset of fewer than *i* variables, and then any additional variable *V*, we can find a value for *V* consistent with the previous values.

The completeness results generally involve obtaining the *i*-closure, for an appropriate *i*, of the clique tree representation of the direct k-tree constraint network for a CSP. If we wish we can go a step further, adding the original variables back in, reducing their domains of values as required by the constraints (e.g. through consistency propagation as in [Freuder, 78]). We thus finish with a (non-binary) constraint network on the original variables, with the desired completeness property.

This final network can be visualized in two ways. All constraints (including the variables viewed as unary constraints) can be represented by vertices. Constraints which share variables are linked by edges at least to the

extent needed to ensure that a variable cannot be assigned two different values at the same time to satisfy two constraints. This representation is descended from the constraint network representation I presented in [Freuder 78], and which I view here as a form of metaproblem representation. Alternatively higher order constraints can be viewed as hyperedges in a hypergraph representation [Dechter and Pearl, 89].

We need to know that all this transforming never changes the set of solutions for the original problem variables; however, this is fairly obvious. Similar transformations have been used before in the literature. The clique tree representation incorporates the original constraints, so no new solutions can creep in. The new constraints only state explicitly subproblems of the original problem, and ensure that two subproblem solutions do not assign different values to the same variable; thus no old solutions are ruled out. K-consistency can be used as a preprocessing step in solving CSPs. It only removes inconsistent values. Adding the original variables back only allows established constraints to reflect explicitly restrictions on permissible values that they already express implicitly.

3 MULTI-CLIQUE COMPLETABILITY

This section contains the main result of the paper. A set of variables for a CSP will be called a *clique* if the vertices corresponding to the set of variables, together with the constraints between pairs of these variables, constitute a clique in its direct constraint network representation. A *j-clique* of variables is a clique containing *j* variables. A CSP representation is *i multi-clique completable* iff any solution for a subset of variables belonging to any *i* (or fewer) cliques is completable.

The definition allows us to pull, for example, two values from one clique, five from another, all the values from a third. Any cliques will do, though of course they will all be contained in some maximal clique, which in the case of k-tree structure means some $k+1$ -clique. Note that *i* multi-clique completable implies in particular that a solution for any *i* variables is completable.

The basic idea is to add explicit constraints to the original representation of the CSP sufficient to rule out partial solutions which cannot be completed. This can always be done simply by finding all the complete solutions. The trick is to see if we can get by with less complete processing of the problem. K-tree structures permit that.

Some intuition as to the basic nature of the processing can be gained by thinking of the problem as follows: Given values at a set of cliques in the clique tree we have to a) fill in consistent values in the area "bounded" by those cliques and b) fill in values at the cliques in the subtrees outside this bounded area. The subtrees can be handled recursively. The bounded area can be filled in using the strong *i+1*-consistency of the *i+1*-closure of the clique tree.

A primary use of $i+1$ -consistency here is analogous to the use of 3-consistency to fill in values along a path between two constraint network vertices. It has been recognized that 3-consistency corresponds to "path" consistency; if given values for two variables we can find a value for a third variable, we can find values all along a path between them, such that all the constraints represented by the edges of the path are satisfied [Montanari 74]. I suggest here a generalized "physical" implication of i -consistency, which might be termed *tree consistency*: i -consistency guarantees that given values for $i-1$ variables we can find values for all variables in a tree embedded in the constraint network with these $i-1$ variables as leaves, such that all the binary constraints represented by the edges of this tree will be satisfied.

Theorem 1: Given a k -tree CSP with n variables, each with at most d values, in $O((nd^{k+1})^{i+1})$ time we can obtain an i multi-clique completable representation.

Proof. We obtain the $i+1$ -closure of a clique tree representation of the k -tree direct constraint network of the given CSP. We want to show that this representation is i multi-clique completable. We have seen that the original problem, the clique tree, and the $i+1$ -closure of the clique tree, all have the same set of solutions. When we are done we can add back vertices corresponding to the original variables if we wish, as suggested in Section 2.3.

We are given a solution for a set of variables chosen from i chosen cliques. I outline an algorithm for completing the solution, finding consistent values for the remaining variables. The algorithm recursively processes the representation. The emphasis is on demonstrating that appropriate values must exist; once we know that a value exists it can always be found, if necessary by exhaustive search. There are two cases in the terminal or basis step of the recursion, and two cases in the non-terminal or recursive step.

Note that in choosing new values we can use the induced constraints in the $i+1$ -closure of the clique tree to insure consistency. However, at the same time we can take advantage of the fact that to form a solution it is sufficient that a set of values satisfy the constraints in the original clique tree.

Preprocessing Step:

If there are any cliques for which some variables, but not all, have been assigned values, assign consistent values to the remaining variables in those cliques. This can be done as the original values chosen must have satisfied the constraint involving all the variables from those cliques induced by $i+1$ -closure of the clique tree.

Terminal Step:

Case 1: All the variables at all the cliques have been assigned consistent values.

Case 2: The variables at only one clique have been

assigned consistent values. Make that clique the root of the clique tree. Now 2-consistency guarantees that we can find consistent values at the remaining vertices of the tree, e.g. while traversing the tree in breadth-first order.

Non-terminal Step:

Case 1: At least one of the cliques, C , whose variables have been assigned consistent values is not a leaf of the clique tree. Removing C disconnects the tree into two subtrees, involving two sets of cliques, R and S . The problem we are currently dealing with can be solved by independently and recursively solving two subproblems, involving R plus C and S plus C . We recurse on the subgraphs of the $i+1$ -closure of the clique tree induced by the union of R and C and the union of S and C . The only "communication" in the original clique tree between the cliques of the two subproblems is through C ; if we solve the subproblems independently and the solutions agree on C we will have a solution to the full problem.

Case 2: All of the cliques whose variables have been assigned consistent values are leaves of the clique tree. Since not all cliques contain variables with chosen values, there must be a leaf, L , whose one adjacent vertex, P , does not contain variables with chosen values. We use $i+1$ -consistency to move inward in the tree from L . Choose a metavariable for P , i.e. values for all the variables in the clique P , consistent with all the chosen values at the leaves. From this point we can ignore L ; it's only contact with the other cliques, in the original clique tree, is through P . Recurse on the representation obtained by removing L .

Recursion in either case results in smaller problems; eventually one of the terminal cases must be reached.

The complexity bound largely follows from the bound on Cooper's k -consistency algorithm [Cooper 89], which is $O(n^k d^k)$ for a CSP with n variables of d values. Our metavariables have at most d^{k+1} values, and there will be $n-k$ $k+1$ -clique metavariables, see [Beineke and Pippert 71]. Q.E.D.

It may be more natural to talk directly about completing solutions for sets of variables rather than in terms of cliques. Since any variable is in some clique we have the following corollary (in this and succeeding theorems n will represent the number of variables, d the maximum number of values for any variable):

Corollary 1: For a k -tree CSP, in $O((nd^{k+1})^{i+1})$ time we can obtain a representation in which any solution for any i variables is completable.

In choosing to achieve i multi-clique completable for a particular i there is clearly a tradeoff between computational effort and level of completable. The level and form of completable appropriate for an individual problem will balance the needs of the user against the computational realities.

4 EXTREMES

This section focusses on i multi-clique completability representations of k -tree CSPs for extreme values of i and k : $i=1$ and $n-k$ (the number of maximal cliques), and $k=1$ (the usual tree structures). (For n vertices the maximum k value is n , associated with a complete graph) Improved completability bounds are obtained for the minimal i and k values.

4.1 CLIQUE COMPLETABILITY

A CSP representation is *clique completable* iff any solution for any subset of variables from any one clique is completable.

Theorem 2. A clique completable representation of a k -tree CSP can be obtained in $O(nd^{k+2})$ time.

Proof: This is basically Theorem 1 for $i=1$. However, for tree-structured CSPs 2-closure can be obtained in time linear in the number of variables and quadratic in the maximum number of values for a variable [Dechter and Pearl 88a]. The latter factor comes from the need to pair all the values in one variable with all those in an adjacent variable. However, we have a special case here. Adjacent metavariables in the clique tree share k out of $k+1$ variables from the original problem. Assuring consistency between them amounts to establishing a $k+2$ -ary constraint in the original problem, or solving a CSP with $k+2$ variables. This can be done in $O(d^{k+2})$ time (for fixed k). If we do this for pairs of adjacent metavariables starting at each leaf and moving up to the root, then going back down from the root to each leaf, we achieve 2-consistency in $O(nd^{k+2})$ time. Q.E.D.

It is easy to see, using results from [Freuder 90] and the k -tree literature, that achieving $k+1$ -consistency for k -trees would make all clique solutions completable. As we have seen a k -tree can be built by starting with a $k+1$ -clique and adding vertices which each in turn link to all the vertices in an already present k -clique, forming another $k+1$ -clique. The initial clique is called a *basis*, and any k -clique can serve as a basis for building the k -tree in this way [Proskurowski 84]. Thus to complete a solution for a clique, we could, if we had $k+1$ -consistency, first add values as needed to form the solutions to a k -clique, then choose values for each variable we added in turn as we built the k -tree. At each point we would only need to worry about the consistency of the new value with k already chosen values, and $k+1$ -consistency would take care of that. (If we are given all the values for a $k+1$ -clique to begin with, one of the non-basis values will be predetermined.)

However, for $k+1$ -consistency we have a complexity bound $O(n^{k+1}d^{k+1})$. Comparing with the bound from Theorem 2, essentially we trade a factor of n^k against a

factor of d . Clearly as a function of the number of problem variables, n , the bound of Theorem 2 is superior.

Corollary 2: A variable completable representation of a k -tree CSP can be obtained in $O(nd^{k+2})$ time.

Proof. Since every variable is in some clique, this is a special case of clique completable. Q.E.D.

4.2 FULL COMPLETABILITY AND TOTAL-CLIQUE COMPLETABILITY

A CSP representation is *fully completable* iff any solution for any subset of variables is completable. We will see that full completable corresponds to $n-k$ multi-clique completable.

Theorem 3: A fully completable representation of a k -tree CSP can be obtained in $O((nd^{k+1})^{n-k+1})$ time.

Proof: As observed earlier, there are $n-k$ maximal cliques in a k -tree. Any variable must be in at least one of these. Thus $n-k$ multi-clique completable implies full completable; it can be obtained in $O((nd^{k+1})^{n-k+1})$ time by Theorem 1. Q.E.D.

A potentially better bound can be obtained for a limited form of $n-k$ multi-clique completable. Define a representation to be total-clique *completable* iff any solution for a subset of variables that constitutes *all* the variables from any set of maximal cliques is completable.

Theorem 4: Given a clique tree representation of a k -tree CSP, with L leaves, we can obtain a total-clique completable representation in $O((nd^{k+1})^{L+1})$ time.

Proof. Observe that in the proof of Theorem 1, after the preprocessing step, we only need more than 2-consistency to deal with the leaf cliques or some subset of them. If all the variables in the cliques involved have been assigned variables to start with we do not need the preprocessing step. Thus we will never need more than $L+1$ -consistency, where L is the number of leaves in the clique tree, regardless of what i is. Q.E.D.

Note that this result places a premium on finding a clique tree with a minimal number of leaves. The same CSP can have two clique tree representations with very different numbers of leaves. In general a k -tree can have approximately $k/(k+1)$ leaves, so the upper bound outlook is not good: on the other hand, there are "chain" clique trees of arbitrary size that have only two leaves. If the clique tree is a *chain*, a tree with a single branch, we can obtain total-clique completable in $O(nd^{k+1})^3$ time.

4.3 TREES

When $k=1$ we have ordinary tree structures. For tree CSPs, Corollary 2 says that we can obtain a variable completable representation in $O(nd^3)$ time. The experienced reader may observe here that 2-closure of the

tree structured constraint network itself, *not* the clique tree, is sufficient to guarantee variable completable, and can be obtained in $O(nd^2)$ time. In fact, for $k=1$, clique completable is Montanari's minimality, and, as Montanari noted, 2-closure of a tree structure is sufficient to achieve minimality [Montanari 74].

Montanari has a further result. He shows that values for any two variables in the 3-closure of a tree CSP, not necessarily from the same clique in the tree (i.e. not necessarily joined by an edge) can be extended to a solution [Montanari 74]. 3-consistency can be obtained in $O(n^3d^3)$ time [Cooper 89]. Furthermore Montanari's method, which first fills in a path between the variables, motivating the approach of Theorem 1, appears to also achieve 2 multi-clique completable (which could extend the solution of as many as four variables).

For $k = 1$ Theorem 1 gives a bound of $O(n^3d^6)$; avoiding the clique tree transformation gives a better bound. Furthermore, we can use the generalization of path consistency to tree consistency suggested in Section 3 to handle i variables, or cliques. This approach requires $i+1$ -consistency, where we have an $O((nd)^{i+1})$ bound. This compares with the $O((nd^2)^{i+1})$ bound given for i multi-clique consistency for $k = 1$ by Theorem 1. We have sketched out a proof of the following theorem, generalizing Montanari's results:

Theorem 5. The $i+1$ -closure of a tree CSP is i multi-clique completable. In particular a solution for any i variables is completable.

The question now arises as to whether these results imply a more efficient alternative to the methods of Theorem 1 for $k > 1$. Theorem 1 already is descended from Montanari's methods. However, it may be possible to extend those methods more directly than Theorem 1 does, perhaps by using the concept of a k -path [Beineke and Pippert 71], or a tree of k -cliques.

5 PARTIAL K-TREES

This section indicates how the previous completable results extend to partial k -trees via their embedding in a k -tree. The embedding transformation again does not change the set of solutions; it involves adding trivial constraints. Partial k -trees allow us to formulate results for general CSPs and for classes of special interest, series-parallel and bandwidth- k CSPs.

Recall that a partial k -tree is a subgraph of a k -tree. As noted above, by adding trivial constraints any constraint network can be made into, or *imbedded* in, a k -tree, for some k . If necessary we can simply add enough trivial constraints to form a complete graph (all possible edges): complete graphs are k -trees. Thus since any constraint network is a partial k -tree for some k , completable results can be provided for *any* CSP. However, if indeed

the embedding is the complete graph, the completable results will not be very interesting.

Combining these observations with Theorem 1 gives us:

Theorem 6: Given an arbitrary constraint network G and an embedding of G in a k -tree, in $O((nd^{k+1})^{i+1})$ time we can obtain an i multi-clique completable representation.

It is obviously desirable to obtain an optimal embedding in the sense of one in which k is as small as possible. An optimal embedding may be hard to find. There is an $O(n^{k+2})$ algorithm to find the embedding [Arnborg, Corneil and Proskurowski 87]. For a given CSP, even the optimal k may be close to n , resulting in a k exponent in the complexity bounds that is close to n .

However, if a 2-tree embedding exists it can be found in linear time [Wald and Colbourn 83]. The class of partial 2-trees is the class of series-parallel networks, which are of special interest in engineering. Thus we have:

Theorem 7: Given a series-parallel constraint network: in $O((nd^3)^{i+1})$ time we can obtain an i multi-clique completable representation.

For larger optimal k , heuristic methods that settle for suboptimal embeddings may prove effective. Dechter and Pearl's work on tree-clustering [Dechter and Pearl 89] provides a closely related representation that should be useful in treating arbitrary graphs. In some cases the embedding may be obvious from the semantics. Consider, for example, the scheduling problem in Section 1.1. Each event can constrain up to 5 succeeding events. The embedding into a 5-tree is not hard to find: add trivial constraints as needed so that each event does constrain the 5 succeeding ones.

The direct constraint network of this scheduling problem is a bandwidth-5 graph. A *bandwidth-k graph* is one for which an ordering of the vertices exists with the property that no vertex shares an edge with another more than k vertices before or after it in the ordering. This ordering, call it a *bandwidth-k ordering*, can be found, if it exists, in polynomial time for fixed k [Saxe 80]. Bandwidth has been studied in connection with CSPs [Zabih 90]. Restricting problem structure to fit a bandwidth- k criterion would seem to be a natural way of specifying representational restrictions for classes of CSP problems in, for example, temporal reasoning.

Theorem 8. Given a bandwidth- k ordering of a direct constraint network of a CSP, a representation in which a solution for any set of $k+1$ or more consecutive variables in the ordering is completable can be obtained in $O((nd^{k+1})^3)$ time.

Proof. Bandwidth- k graphs are easily seen to be partial k -trees; they can be embedded in k -trees by adding trivial constraints as needed to ensure that each vertex is linked to all vertices not more than k vertices away in the ordering.

Moreover, such a k -tree has a clique tree which is a simple chain with two leaves. Any set of $k+1$ or more consecutive values will constitute all the variables in a set of adjacent cliques in this chain. Apply Theorem 4. Q.E.D.

This theorem establishes the $O(n^3)$ result claimed for the scheduling problem in Section 1.1.

6 CONCLUSION

6.1 ILLUSTRATION

A final example will illustrate various forms of completable. Figure 4 can be viewed as representing another simple graph coloring problem. Suppose that there are three colors available at each vertex: red, blue and green. The graph to be colored, and the 2-tree direct constraint network for this coloring problem, both have the structure shown in the figure.

The clique tree for the 2-tree with the structure shown in Figure 5 is 2-consistent without any further processing. Thus it is clique completable. If we color 1 red, 3 blue and 5 green, for example, we can go on to find consistent colors for 2 (green), 4 (red) and 6 (blue).

However, the representation is not 2 multi-clique completable. Suppose, for example, we choose blue for 3, red for 5 and red for 6. These values do not violate any of the constraints embodied explicitly in the clique tree. However, there is no complete solution to the problem which includes this assignment of values.

The 3-closure of the clique tree is 2 multi-clique completable. Obtaining the 3-closure will, in particular, induce a new constraint, C, between the 2,4,6 vertex and the 1,3,5 vertex. The values blue for 3 and red for 5 appear in only one triple at the 1,3,5 vertex: g,b,r. The value red for 6 appears in two triples at the 2,4,6 vertex: g,b,r and b,g,r. However, the new constraint C does not allow either 2,4,6 triple to be paired with the 1,3,5 triple. Thus the incompletable combination of blue for 3, red for 5, and red for 6 is not a solution for the subset of variables 3, 5 and 6 in the 3-closure of the clique-tree.

Since the clique tree has only two leaves, the 3-closure is total-clique completable. Now for this small sample problem it might have been easier simply to find all the complete solutions. However, we could make the coloring problem considerably larger and more complex while still retaining a simple chain clique tree structure.

6.2 SUMMARY

For k -tree CSPs a representation can be obtained in $O((nd^{k+1})^{i+1})$ time in which a solution for variables taken from any i cliques of the direct constraint network can always be extended to a complete solution to the problem. This is called an i multi-clique completable representation. The representation obtained is the i -closure

of the clique tree of the direct constraint network of the k -tree CSP. Completable representations for any CSP can be found by first embedding the CSP constraint network in a suitable k -tree.

Tighter bounds than those derivable in the obvious way from i multi-clique completable are available for clique completable, variable completable and for trees (1-trees). The bounds on full and total-clique completable are better than one might expect.

Specific results are available for CSPs with series-parallel and bandwidth- k structure. There is reason to believe that temporal reasoning and scheduling problems, in particular, may admit computationally tractable completable results. Constraint knowledge bases might be preprocessed to provide completable properties.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IRI-8913040. The Government has certain rights in this material. The author is currently a Visiting Scientist at the MIT Artificial Intelligence Laboratory.

References

- [Arnborg, Corneil and Proskurowski 87] Complexity of finding embeddings in a k -tree. *Siam J. Alg. Disc. Meth.* 8, 2, 277-284.
- [Beineke and Pippert 71] Properties and characterizations of k -trees. *Mathematika* 18, 141-151.
- [Cooper 89] An optimal k -consistency algorithm. *Artificial Intelligence* 41, 89-95.
- [Dechter 90] From local to global consistency. *Proc. of the Eighth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*. Morgan Kaufmann.
- [Dechter, Meiri and Pearl 89] Temporal constraint networks. In Brachman, Levesque and Reiter (eds.) *Principles of Knowledge Representation and Reasoning: Proceedings of the First International Conference*. San Mateo, CA: Morgan Kaufmann.
- [Dechter and Pearl 88a] Network-based heuristics for constraint-satisfaction problems, *Art. Int.* 34(1).
- [Dechter and Pearl 88b] Constraint-directed approach to diagnosis. UCLA CS Dept. Tech. Rept. R-72-1.
- [Dechter and Pearl 89] Tree clustering for constraint networks. *Artificial Intelligence*, 353-366.
- [de Kleer 89] A comparison of ATMS and CSP techniques. *Proc. of the Eleventh International Joint Conference on Artificial Intelligence*, 290-296.
- [Freuder 78] Synthesizing constraint expressions. *CACM* 21, 11.

- [Freuder 82] A sufficient condition for backtrack-free search. *JACM* 29, 1.
- [Freuder 85] A sufficient condition for backtrack-bounded search, *JACM* 32, 4.
- [Freuder 89] Partial constraint satisfaction. *Proc. of the Eleventh International Joint Conference on Artificial Intelligence*, 278-283.
- [Freuder 90] Complexity of k-tree structured constraint satisfaction problems. *Proc. of the Eighth National Conference on Artificial Intelligence*, AAAI Press, Menlo Park, CA, .
- [Kuipers and Berleant 88] Using incomplete quantitative knowledge in qualitative reasoning. *Proc. of the Seventh National Conference on Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA.
- [Mackworth 77] Consistency in networks of relations. *Artificial Intelligence* 8, 99-118.
- [Montanari 74] Networks of constraints: fundamental properties and applications to picture processing. *Information Sciences* 7, 95-132.
- [Proskurowski 84] Separating subgraphs in k-trees: Cables and caterpillars, *Discrete Math.* 49, 275-285.
- [Reiter and Mackworth 88] A logical framework for depiction and image interpretation. Univ. of British Columbia Dept. of CS Tech. Rept. 88-17.
- [Robertson and Seymour 86] Graph minors II: algorithmic aspects of tree-width. *Journal of Algorithms* 7, 309-322.
- [Rossi, Dhar and Petri 89] On the equivalence of constraint satisfaction problems. MCC Technical Report ACT-AI-222-89. MCC, Austin, Texas 78759.
- [Saxe 80] Dynamic programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM J. Alg. Disc. Meth.* 1, 363-369.
- [Tambe and Rosenbloom 90] A framework for investigating production system formulations with polynomially bounded match. *Proc. of the Eighth National Conference on Artificial Intelligence*, AAAI Press, Menlo Park, CA, 693-700.
- [van Beek 89] Approximation algorithms for temporal reasoning. *IJCAI-89*.
- [Vilain, Kautz and van Beek 89] Constraint propagation algorithms for temporal reasoning: a revised report. in *Readings in Qualitative Reasoning About Physical Systems*. de Kleer and Weld, editors. Morgan Kaufmann.
- [Wald and Colbourn 83] Steiner trees, partial 2-trees, and minimum IFI networks. *Networks* 13, 159-167.
- [Wimer 87] Linear algorithms on k-terminal graphs. Ph.D. dissertation, Clemson.
- [Zabih 90] Some applications of graph bandwidth to constraint satisfaction problems. *Proc. of the Eighth National Conference on Artificial Intelligence*, AAAI Press, Menlo Park, CA, 46-51.

A General Framework for Modal Deduction

Alan M. Frisch and Richard B. Scherl

University of Illinois

Department of Computer Science and Beckman Institute

405 North Mathews

Urbana, Illinois 61801

frisch@cs.uiuc.edu scherl@cs.uiuc.edu

Abstract

A general method of automated modal logic theorem proving is discussed and illustrated. This method is based on the substitutional framework for the development of systems for hybrid reasoning. Sentences in modal logic are translated into a constraint logic in which the constraints represent the connections between worlds in the possible world semantics for modal logic. Deduction in the constraint logic is performed by a non-modal deductive system which has been systematically enhanced with special-purpose constraint processing mechanisms. The result is a modal logic theorem prover, whose soundness and completeness is an immediate consequence of the correctness of the non-modal deductive system and some general results on constraint deduction. The framework achieves significant generality in that it provides for the extension of a wide range of non-modal systems to corresponding modal systems and that this can be done for a wide range of modal logics.

1 Introduction

A wide range of modal logics have been finding applications in many areas of AI research and recently a number of automated deduction methods for modal logics have been developed. A number of these methods utilize ordinary non-modal inference rules augmented with a special purpose unification procedure that reasons about the accessibility of possible worlds. The deduction methods that fall into this class have been based on a variety of non-modal methods including resolution [Ohlbach, 1988, Auffray and Enjalbert, 1989], sequent calculus [Jackson and Reichgelt, 1989] and the matrix method [Wallen, 1990]. Additionally, outside of the methods

designed specifically for automation, Fitting[1983] has developed a related approach based on deductive tableaux. This leads one to wonder whether there are any methods for systematically constructing modal deduction systems from non-modal systems, whether an understanding of the modal systems can be obtained from an understanding of the original non-modal systems, and whether there is a common set of principles underlying systems constructed in this manner.

This paper presents a framework for the *systematic* design of modal deduction systems that allows us to respond affirmatively to each of the above questions. The framework achieves significant generality in that it provides for the extension of a wide range of non-modal deduction systems to corresponding deduction systems for a wide range of modal logics. Along with this transformation of the deductive rules, the proofs of completeness of the original deductive system are transformed into a proof of completeness of the new deductive system. The resulting description and proof of correctness of the new system is relatively simple because it draws upon the general principles of the framework. The framework enables us to provide a proof of completeness of a deductive method, based upon resolution and paramodulation, for modal logics with equality. Most of the features of most of the members of the class of methods mentioned above can be recast into this framework. Those aspects of the methods that can not be so reconstructed serve to delimit the domain of applicability of the framework. A deeper understanding of how this class of related approaches work has been achieved.

Very roughly stated, our method works as follows: We start with a set of modal sentences on which the deduction is to be performed. These sentences are translated into non-modal sentences that contain K -atoms, atoms whose predicate symbol represents the accessibility relation between possible worlds. In the case of varying domain modal logics, E -atoms, atoms whose predicate symbol denotes the existence

of an entity in a world, are also needed. Deduction on the resulting sentences is performed by a non-modal deduction system that is extended with special purpose built-in reasoners that handle the K - and E -atoms.

Now let us look at the method more closely. A modal deduction system produced by our method operates on non-modal sentences of a language that we call constraint logic. The sentences of constraint logic are like sentences of ordinary first-order logic except that tuples of variables may be associated with constraints that restrict the domain over which the variables range. Sentences of constraint logic are syntactic variants of ordinary first order sentences since the constraints can be translated out. The name constraint logic is chosen because the logic can be seen as a generalization of Horn-clause constraint logic programming [Jaffar and Llasses, 1987, Van Hentenryk, 1989] to full first-order logic.

The constraint logic sentences are obtained by translating the modal sentences on which one wishes to perform deduction. Based on the possible-worlds semantics of modal logic, a modal sentence can be viewed as implicit discourse about possible worlds and the accessibility relation between them. Any modal sentence is thus equivalent (in a certain sense) to a non-modal sentence that explicitly discusses possible worlds and their inter-accessibility. Our translation of modal sentences to constraint sentences accords the accessibility relation a special status in the constraint logic.

The deductive system that reasons in the constraint logic is a hybrid system that combines ordinary deduction with special purpose methods for constraint processing. The construction of this deductive system is carried out within the substitutional framework [Frisch, 1991] for the development of systems for hybrid reasoning. In this approach, a set of predicates are designated as constraint atoms. In the case of modal inference, the only predicate symbols in the designated set are K and E .

The input sentences to a hybrid reasoning system constructed within the substitutional framework are divided into two groups. One group of sentences, the S -sentences, contain no predicates other than those designated as constraint predicates. The S -sentences together form a theory called Σ . The other sentences, the A -sentences, are the constrained sentences. In the A -sentences, the predicates from the designated set occur only in the constraints. The only predicates that occur in the constraints are those from the designated set. In the modal case, the constraints on the A -sentences are conjunctions of K - and E -atoms.

The substitutional framework provides a method for transforming an ordinary deductive system into a

hybrid deductive system in which the set of designated predicates are handled by a special-purpose reasoner. The overall deductive mechanism that reasons with the A -sentences invokes the special constraint processing mechanism to reason with the constraint predicates. All reasoning about the predicates from the designated set is done by the special purpose constraint processing mechanism.

The resolution and paramodulation rules of inference are used to illustrate the method by which a non-modal deductive system can be converted systematically to a constraint-deduction system that deals with constraints made up of K and E -atoms. So the final result will be closest to the resolution-based modal deduction systems of Ohlbach [1988] and of Auffray and Enjalbert [1989] and the development here of a resolution-based modal deductive method can be viewed as a reconstruction of these systems.

The framework is limited to logics whose possible world semantics restrict the accessibility relation to serial relations—some world is accessible from every world. The logics covered are those characterized by any combination of the reflexive, symmetric, transitive and Euclidian restrictions on the accessibility relation. Given the restriction to serial logics, this yields 9 distinct modal logics, including D , which is the minimal serial modal logic, T , for which the accessibility relation is required to be reflexive, $S4$, whose accessibility relation is required to be transitive as well as reflexive, and $S5$, which has an accessibility relation that is symmetric in addition to being reflexive and transitive.

In the course of this article, we illustrate the method being developed with a very simple version of the wise-man puzzle [Genesereth and Nilsson, 1987], which is commonly used as a test problem for formalisms involving knowledge and belief. The story is that there are some number of wise men who are told by the king that at least one of them has a white spot on his forehead. In reality, all of the wise men have white spots on their foreheads. Each wise man can see the others' foreheads, but not his own. In this version of the puzzle, there are two wise men named A and B . (1) A knows that if A does not have a white spot, B will know that A does not have a white spot. (2) A knows that B knows that either A or B has a white spot. B says that he does not know whether he has a white spot, and (3) A thereby knows that B does not know whether he has a white spot. The problem is to prove that (4) A knows that he has a white spot.

In a modal logic formulation of this puzzle, the modal operator \Box is used to represent knowledge. It is necessary to introduce a distinct modal operator for each agent, which we call \Box_A and \Box_B . The semantics must also be generalized to include an ac-

- | |
|---|
| (1) $\square_A (\neg \text{White}(A) \rightarrow \square_B (\neg \text{White}(A)))$ |
| (2) $\square_A (\square_B (\text{White}(A) \vee \text{White}(B)))$ |
| (3) $\square_A (\neg \square_B (\text{White}(B)))$ |
| (4) $\neg \square_A (\text{White}(A))$ |

Figure 1: Wise Man Puzzle

cessibility relation for each of the modal operators. Figure 1 contains the representation of this puzzle in modal logic. The puzzle is formulated for a refutational proof procedure with the last line being the negation of the hypothesis to be proven. We will see that the minimal serial modal logic D suffices to obtain a solution to this puzzle.

Section 2 covers the procedure for translating modal sentences to constrained sentences in prenex normal form. In Section 3, the substitutional framework for the construction of systems of hybrid reasoning is used to modify ordinary resolution to obtain a form of resolution for constrained clauses. Section 4 covers a specialized algorithm for modal constraint processing. Sections 2 through 4 assume the modal logic is a constant domain modal logic. Section 5 briefly discusses how the framework can handle varying domain modal logics. In the conclusion, Section 6, future research is discussed.

2 Translation to Constrained Clauses

The first part of this section describes a procedure for translating sentences in a constant domain modal logic into a non-modal language with constrained quantifiers. We call the resulting set of sentences a *reified modal logic*. The second part of the section describes a procedure for converting these sentences into Skolem normal form. As both of the translations preserve satisfiability, the initial set of modal sentences is satisfiable if and only if the resulting set of Skolem normal form sentences of reified modal logic is satisfiable.

The translation from modal logic to reified modal logic introduces the predicate K as a constraint on variables that range over possible worlds. The constraints on sentences of the reified modal logic represent the connections between possible worlds in the Kripke semantics for the modal logic. Sentences in modal logic are evaluated in a *model structure* [Kripke, 1963], which is a system of interconnected elements called possible worlds with one of the possible worlds designated as the actual world. If a and b are two worlds, $K(a, b)$ means that b is accessible from a . Then, every proposition true in b is possible in a . A proposition is necessary in world a , if it is true in every world accessible from a .

For example, $\square P$ in a propositional modal logic is translated into the constrained sentence $\forall z: K(0, z) \rightarrow P(z)$, which is an abbreviation for $\forall z K(0, z) \rightarrow P(z)$. This sentence can be read as saying for all worlds z , if z is accessible from world 0, which denotes the actual world, then P is true in z . Similarly, $\diamond P$ is translated into the constrained sentence $\exists z: K(0, z) \wedge P(z)$, which is an abbreviation for $\exists z K(0, z) \wedge P(z)$. This sentence can be read as saying that there is some world z that is connected to the actual world 0 and P is true in z .

The translation function T_0 maps sets of modal sentences to sets of constrained sentences while preserving satisfiability. The translation function at an arbitrary step is represented by T_i where $i \geq 0$. The integer i is initially set to 0 and is incremented at various points in the translation process. The translation function is recursively defined as follows:

- $T_0(S) = \{T_0(s) | s \in S\}$, where S is a set of sentences.
- $T_i(z) = z'$, where z is a variable.
- $T_i(f(t_1, \dots, t_n)) = f'(T_i(t_1), \dots, T_i(t_n), w_i)$ where $f(t_1, \dots, t_n)$ is a term
- $T_i(a) = a'(w_i)$, where a is a constant.
- $T_i(P(t_1, \dots, t_n)) = P'(T_i(t_1), \dots, T_i(t_n), w_i)$ where P is a predicate.
- $T_i(\varphi \wedge \psi) = T_i(\varphi) \wedge T_i(\psi)$
- $T_i(\neg \varphi) = \neg T_i(\varphi)$
- $T_i(\varphi \vee \psi) = T_i(\varphi) \vee T_i(\psi)$
- $T_i(\forall z \varphi) = \forall z' T_i(\varphi)$
- $T_i(\exists z \varphi) = \exists z' T_i(\varphi)$
- $T_i(\square \varphi) = \forall w_{i+1}: K(w_i, w_{i+1}) T_{i+1}(\varphi)$
- $T_i(\diamond \varphi) = \exists w_{i+1}: K(w_i, w_{i+1}) T_{i+1}(\varphi)$

In the sentences that result from the translation, w_0 is a constant symbol which denotes the actual world, but it is replaced by 0 to distinguish it clearly from the other w_i , each of which is a variable. The translation function is similar to those proposed by Moore[1980] and Morgan [1976].

The constraint theory Σ must contain the axioms that define the conditions on the accessibility relation for the modal logic being used. For each of the logics under consideration here, a subset of the following axioms is needed:

- Serial: $\forall w_1 \exists w_2 K(w_1, w_2)$
- Reflexive: $\forall w_1 K(w_1, w_1)$
- Euclidean: $\forall w_1, w_2, w_3 K(w_1, w_2) \wedge K(w_1, w_3) \rightarrow K(w_2, w_3)$
- Symmetric: $\forall w_1, w_2 K(w_1, w_2) \rightarrow K(w_2, w_1)$
- Transitive: $\forall w_1, w_2, w_3 K(w_1, w_2) \wedge K(w_2, w_3) \rightarrow K(w_1, w_3)$

The correctness of the translation function T_0 is established by the following theorem.

Theorem 1 Let \mathcal{L}_m be a modal logic and let \mathcal{A} be the axioms that define the conditions on the accessibility relation of the possible world semantics for \mathcal{L}_m . For any set of modal sentences α of \mathcal{L}_m , α is satisfiable iff $T_0(\alpha) \cup \mathcal{A}$ is satisfiable.

Now that the modal sentences have been translated to constrained sentences, we turn to the second topic of the section—the translation of constrained sentences to constrained clauses while preserving satisfiability. The first step is to put the constrained sentences in prenex form. The following theorem establishes that any constrained formula may be rewritten to a logically equivalent formula in prenex form by using six equivalence schemata (derived from [Frisch, 1991]).

Theorem 2 (Prenex Transform Theorem)

If φ and ψ are formulas and ψ has no free occurrences of the variable y and t is an arbitrary term, then:

$$\begin{aligned} \neg \exists y. K(t, y) \varphi &\equiv \forall y. K(t, y) \neg \varphi \\ \neg \forall y. K(t, y) \varphi &\equiv \exists y. K(t, y) \neg \varphi \\ (\exists y. K(t, y) \varphi) \wedge \psi &\equiv \exists y. K(t, y) (\varphi \wedge \psi) \\ (\forall y. K(t, y) \varphi) \vee \psi &\equiv \forall y. K(t, y) (\varphi \vee \psi) \end{aligned}$$

Furthermore, if Σ contains the seriality axiom— $\forall w_1 \exists w_2 K(w_1, w_2)$ —then:

$$\begin{aligned} ((\exists y. K(t, y) \varphi) \vee \psi) \cup \Sigma &\equiv \exists y. K(t, y) (\varphi \vee \psi) \cup \Sigma \\ ((\forall y. K(t, y) \varphi) \wedge \psi) \cup \Sigma &\equiv \forall y. K(t, y) (\varphi \wedge \psi) \cup \Sigma \end{aligned}$$

Using these equivalences, the statements produced by the translation function can be put in prenex normal form by moving all quantifiers (constrained and otherwise) to the front.

The next step is the elimination of constrained existential quantifiers through Skolemization. Skolemization of constrained variables occurs as with ordinary variables, except that information must be added to Σ . For example, consider the constrained variable y in $\exists y. K(t, y) P(y)$. The variable y is replaced by some Skolem function, say SK , yielding $P(SK)$. Additionally, $K(t, SK)$ must be added to Σ . All quantifiers in the translated sentences will be of the form $Q_i; z_i: C_i$ where Q_i is either an existential or universal quantifier and C_i is a possibly empty constraint. If the constraint is empty, the variable z_i is an ordinary unconstrained variable. Otherwise, the constraint is $K(t_{i-1}, z_i)$. The general procedure for the Skolemization of constrained variables is given in Figure 2 and its correctness is established by the following theorem.

Theorem 3 (Skolem Transform Theorem)

$\alpha \cup \Sigma$, a set of prenex form constrained sentences, is satisfiable iff its Skolem Normal Transform is.

Input: A set α of prenex form sentences, and a constraint theory Σ .

While α contains a sentence ϕ that contains an existential quantifier do:

Observe that ϕ is of the form

$$\forall z_1: C_1 \dots \forall z_{n-1}: C_{n-1} \exists z_n: C_n M$$

where $n \geq 0$, M is a prenex-form formula, and each C_i is of the form $K(t_{i-1}, z_i)$ (in which case z_i is a constrained variable) or is empty (in which case z_i is an ordinary unconstrained variable).

Choose f , some n-ary function symbol that does not occur in α or Σ and let θ be $\{f(z_1, \dots, z_{n-1})/z_n\}$. Replace ϕ in α by the sentence

$$\forall z_1: C_1 \dots \forall z_{n-1}: C_{n-1} (M\theta)$$

and if C_n is not empty, add

$$\forall z_1 \dots z_{n-1} C_1 \wedge \dots \wedge C_{n-1} \rightarrow (C_n\theta)$$

to Σ .

Figure 2: Constrained Skolemization Procedure

Note also the following property of this procedure.

Proposition 1 When the Skolemization procedure is applied to the constrained sentences translated from modal sentences, all of the clauses added to the constraint theory Σ are definite clauses.

The definite clauses created by the constrained Skolemization procedure are added to the constraint theory Σ along with the axioms that define the restrictions on the accessibility relation in the possible world semantics for the logic under consideration. Since the axioms are all definite clauses, Σ contains only definite clauses. Thus the final result is a set of constrained clauses as well as the definite clause theory Σ .

As an illustration of the entire translation, consider the wise-man puzzle given earlier. The modal system to be used is D —the minimal serial modal logic. In this case, we have to utilize distinct K predicates (K_A and K_B) to represent the accessibility relations corresponding to the modal operators \square_A and \square_B . We treat the names A and B as rigid designators and therefore do not translate them as denoting relative to a possible world—i.e. with an additional argument position. Applying the translation to (1) – (4) yields:

- (5) $\forall w_1:K_A(0, w_1) (\neg \text{White}'(A', w_1) \rightarrow \forall w_2:K_B(w_1, w_2) (\neg \text{White}'(A', w_2)))$
- (6) $\forall w_1:K_A(0, w_1) \forall w_2:K_B(w_1, w_2) (\text{White}'(A', w_2) \vee \text{White}'(B', w_2))$
- (7) $\forall w_1:K_A(0, w_1) \exists w_2:K_B(w_1, w_2) \neg \text{White}'(B', w_2)$
- (8) $\exists w_1:K_A(0, w_1) \neg \text{White}'(A', w_1)$
(negated hypothesis)

The axioms

- (9) $\forall w_1 K_A(w_1, SK'(w_1))$ (serial axiom for K_A)
- (10) $\forall w_1 K_B(w_1, SK''(w_1))$ (serial axiom for K_B)

are added to the constraint theory for this problem, which we call Σ_{2wm} . For simplicity, these have been entered in Skolem Normal form. Translating (5) – (8) into constrained clauses yields (11) – (14) respectively.

- (11) $\forall w_1:K_A(0, w_1) \neg \text{White}'(A', w_1) \rightarrow \forall w_2:K_B(w_1, w_2) (\neg \text{White}'(A', w_2))$
- (12) $\forall w_1:K_A(0, w_1) \forall w_2:K_B(w_1, w_2) (\text{White}'(A', w_2) \vee \text{White}'(B', w_2))$
- (13) $\forall w_1:K_A(0, w_1) \neg \text{White}'(B', Sk(w_1))$
- (14) $\neg \text{White}'(A', Sk_2)$ (negated hypothesis)

Σ_{2wm} contains the following sentences.

- a. $\forall w_1 K_A(0, w_1) \rightarrow K_B(w_1, Sk(w_1))$
(from Skolemization of 7)
- b. $K_A(0, Sk_2)$ (from Skolemization of 8)
- c. $\forall w_1 K_A(w_1, SK'(w_1))$ (from 9)
- d. $\forall w_1 K_B(w_1, SK'(w_1))$ (from 10)

To summarize, the transformations given in this section translate a set of modal sentences into reified modal logic. The reified modal logic that results takes the form of a constraint logic—that is, it consists of a set of constrained formulas and a constraint theory. The next section presents a class of deduction systems that operate on an any constraint logic, regardless of whether the constrained formulas form a reified modal logic.

3 Constraint Deduction

This section presents a general framework for constructing and analyzing a particular class of hybrid deduction systems, called substitutional systems, for constraint logic. This substitutional framework provides a general method for systematically transforming an automated deduction system for FOPC into one for constraint logic, and for transforming the completeness proof of the original deduction system into one for the resulting deduction system. These transformations can be applied to any first-order deduction system that treats variables schematically through the use of unification.

This section presents the major ideas underlying the substitutional framework and states, but does not prove, the fundamental theorems. The theorems presented here generalize, in two ways, those that Frisch [1991] developed for a sorted logic called SFOPC. The atomic constraints in SFOPC are composed of monadic predicates, called sort symbols, applied to variables. So the first generalization here is that atomic constraints may be any atomic formula whose predicate symbol is a constraint symbol. The second generalization is that the language contains the equality symbol, which is precluded from the previous work. The equality symbol is interpreted as a logical symbol denoting identity. It is not a constraint symbol, so it may occur in the body of an A-formula, but it may not appear in any constraints on the formulas of the sort theory.

Space precludes in depth discussion of how the framework can be used to construct a wide range of deductive rules for constraint logic, so this section concentrates on ordinary resolution and paramodulation. To further simplify the presentation we make the assumption, which is not necessary, that all constraints on A-formulas are positive—that is, if they were transformed into conjunctive or disjunctive normal form they would contain no negation signs. As all constraints in reified modal logic are positive, this assumption poses no problems for our present use of constraint deduction. We also assume that all sentences are in Skolem Normal Form. Thus, the words “sentence” and “theory” only refer to sentences and theories in Skolem Normal Form.

An understanding of the role of unification in ordinary deduction forms the basis upon which the substitutional framework is built. Since the introduction of resolution in 1965, nearly every automated deduction system has handled universally quantified variables by using a unification algorithm. By paralleling this general approach to deduction with ordinary variables, the substitutional framework for deduction with constrained variables achieves the same degree of generality.

The gist of this approach is that a quantified sentence is treated as a schema standing for the set of its ground instances. The justification for this is found in Herbrand’s Theorem, which states that a set of quantified sentences is satisfiable if, and only if, the set containing every ground instance of every one of the sentences is satisfiable. Thus, a formula such as $P(z) \vee Q(z)$ can be read either as $\forall z P(z) \vee Q(z)$ or as a schema standing for the set of all its ground instances.

A deduction system that operates on ground sentences can be made to operate on quantified sentences by replacing tests for equality between expressions with tests for unifiability between them. Sometimes this is all that needs to be done, though

other times additional mechanisms must be incorporated. In either case, the idea is that deductions on quantified sentences are themselves schematic for deductions on ground sentences. A schematic deduction is said to *lift* each deduction that is an instance of it.

Given a deduction system based on unification we would like to show that it does indeed treat quantified sentences as schemas. This usually takes the form of a *lifting theorem* for the deduction system stating that every deduction that can be made from the ground instances of a set of sentences can be made schematically from the sentences themselves.

As an example to illustrate these ideas, consider resolving $P(x, a) \vee Q(x, y)$ with $\neg P(a, z)$. Herbrand's Theorem tells us to consider each of these clauses as standing for the set of all its ground instances. Observe that every ground clause that can be obtained by resolving a ground instance of the first clause with a ground instance of the second is an instance of $Q(a, y)$. Through the use of unification, resolution for non-ground clauses is designed so that it produces $Q(a, y)$ when it resolves the two non-ground clauses together. The Lifting Theorem for Resolution states that this relationship is always obtained.

A substitutional system for constraint logic treats quantified sentences as sentence schemas in much the same way that an ordinary deduction system does. The only difference is that a sentence with constraints does not stand for the set of all its ground instances, but only for those obtained by replacing variables with terms that respect the constraints attached to the variables. Whether or not the constraints are respected depends upon the constraint theory Σ . The ground instances of a constrained formula are therefore referred to as Σ -ground instances and defined as such:

Definition 4 Let Σ be a constraint theory, e/C be a constrained expression, and θ be a substitution such that $e\theta$ is ground. Then $e\theta$ is a Σ -ground instance of e/C if $\Sigma \models \exists C\theta$.¹ The set of all Σ -ground instances of e is denoted by $e_{\Sigma gr}$.

Observe that by this definition a ground instance of a constrained expression has no variables and no constraints. Also observe that some constrained expressions, such as $P(x)/Q(x) \wedge \neg Q(x)$, have no Σ -ground instances. We say that a constraint C is Σ -solvable if $\Sigma \models \exists C$. Hence, a constrained expression, e/C , has a Σ -ground instance if, and only if, C is Σ -solvable.

This treatment of restricted quantifiers is justified

¹Here, and in general, an expression of the form $\exists \phi$ denotes the existential closure of ϕ —that is, the formula $\exists z_1 \dots z_n \phi$ where z_1, \dots, z_n are the freely-occurring variables of ϕ .

by the *Constraint Herbrand Theorem*, which follows this paragraph. This theorem states that, under certain circumstances, a constraint theory Σ and a set α of constrained sentences are jointly satisfiable if, and only if, the set of all Σ -ground instances of α is satisfiable. In saying that α and Σ can be replaced by the Σ -ground instances of α , the Constraint Herbrand Theorem justifies using Σ only in instantiating formulas. Notice that Σ must be used in generating the correct instances of α , but once they are obtained Σ is irrelevant because ground instances have no constraints.

Theorem 5 (Constraint Herbrand Theorem) Let α be a set of constrained Skolem normal form sentences with only positive constraints and let Σ be a constraint theory that has a least Herbrand model. Then $\alpha \cup \Sigma$ is unsatisfiable if, and only if, $\alpha_{\Sigma gr}$ is unsatisfiable.

The rules of resolution and paramodulation operate on clauses. Now, since the convention is to drop quantifiers when converting to clause form,

$$\forall w_1:K(0, w_1) \dots \forall w_n:K(w_{n-1}, w_n) \varphi$$

where φ is a disjunction of literals can be written as

$$\varphi / K(0, w_1) \wedge \dots \wedge K(w_{n-1}, w_n).$$

With this convention, after all existential quantifiers have been eliminated, universal quantifiers may be dropped, the remaining formula can be put in conjunctive normal form and separated into clause form. Each separate clause will have a (possibly empty) constraint expression which is a conjunction of K -atoms expressing the constraints on the variables in that clause.

Translating (11) – (14) into constrained clauses yields (15) – (18) respectively.

- (15) $White'(A', w_1) \vee \neg White'(A', w_2) / K_A(0, w_1) \wedge K_B(w_1, w_2)$
- (16) $White'(A', w_2) \vee White'(B', w_2) / K_A(0, w_1) \wedge K_B(w_1, w_2)$
- (17) $\neg White'(B', Sk(w_1)) / K_A(0, w_1)$
- (18) $\neg White'(A', Sk_2))$

Σ_{2um} remains as given earlier.

A deduction system for constraint logic can be produced systematically from an ordinary first-order deduction system by integrating a special-purpose constraint processing mechanism into it. Suppose that the ordinary deduction system has an inference rule of the form

From ϕ_1 and ϕ_2 derive $\phi_3\theta$,

where θ is a substitution used in the deduction. The corresponding constraint logic rule is

From ϕ_1/C_1 and ϕ_2/C_2 derive $(\phi_3/C_1 \wedge C_2)\theta$, provided that $(C_1 \wedge C_2)\theta$ is Σ -solvable.

The provision that $(C_1 \wedge C_2)\theta$ is Σ -solvable ensures that the derived formula has at least one ground instance. Without this provision, the inference rule could be used to produce a proof that is not schematic for any ground proof, thereby compromising soundness. The only place in a substitutional reasoner where Σ is used is in testing this provision.

The above schema for constraint deduction can be instantiated to yield the rules of constraint resolution and constraint paramodulation. Simply consider $\phi_3\theta$ to be a resolvent (paramodulant) of the clauses ϕ_1 and ϕ_2 , where θ is the most general unifier used in performing the resolution (paramodulation).

The justification for handling constraints in this manner parallels that for using unification to handle variables without constraints: Every ground clause that can be obtained by constraint resolution (paramodulation) from a Σ -ground instance of ϕ_1/C_1 and a Σ -ground instance of ϕ_2/C_2 is a Σ -ground instance of a constraint resolvent (paramodulant) of ϕ_1/C_1 and ϕ_2/C_2 . This property is asserted by the following lifting theorem:

Theorem 6 (Lifting Theorem) *Let Σ be a constraint theory, let α be a set of constrained clauses, and let ϕ be a ground clause. If ϕ is derivable from $\alpha_{\Sigma gr}$ by constraint resolution and constraint paramodulation, then some constrained clause that has ϕ as a Σ -ground instance is derivable from α by constraint resolution and constraint paramodulation.*

The proof of a lifting theorem for a constraint deduction system can be systematically produced from a proof of the corresponding theorem for ordinary deduction. The basis for this is that lifting theorems for ordinary deduction systems hold because of certain properties of instantiation, and it can be shown that instantiation for constrained formulas has all these crucial properties.

Putting this all together yields a proof of completeness theorem for constraint resolution and constraint paramodulation.

Theorem 7 (Completeness Theorem) *Let α be a set of constrained clauses and Σ be a constraint theory with a least Herbrand model. If $\Sigma \cup \alpha$ is unsatisfiable then constraint resolution and constraint paramodulation can derive the empty clause from α using the constraint theory Σ .*

If $\Sigma \cup \alpha$ is unsatisfiable, then, by the Constraint Herbrand Theorem, so is $\alpha_{\Sigma gr}$. When operating on

clauses that contain no constraints, constraint resolution (paramodulation) is identical to ordinary resolution (paramodulation), and therefore constraint resolution (paramodulation), like ordinary resolution (paramodulation), can derive the empty clause from $\alpha_{\Sigma gr}$. Finally, the Lifting Theorem enables us to conclude that constraint resolution and constraint paramodulation can derive the empty clause from α using the constraint theory Σ .

The following refutation of clauses (15)–(18) using the constraint theory Σ_{2wm} illustrates the use of constraint resolution.

- (19) $\neg White'(A', w_2)/K_A(0, Sk_2) \wedge K_B(Sk_2, w_2)$
(from 15,18)
- (20) $\neg White'(A', w_2)/K_B(Sk_2, w_2)$
(simplification of 19)
- (21) $White'(B', w_2)/K_B(Sk_2, w_2) \wedge K_A(0, w_1)$
 $\wedge K_B(w_1, w_2)$ (from 16,20)
- (22) $\Box/K_B(Sk_2, Sk(w_4)) \wedge K_A(0, w_1)$
 $\wedge K_B(w_1, Sk(w_4)) \wedge K_A(0, w_4)$
(from 21,17)

At each step it is necessary to verify that the new constraint is Σ -solvable, thereby eliminating some potential resolvents. For example, to resolve clause (18) with the first literal of clause (16) it is necessary to test the constraint $K_A(0, w_1) \wedge K_B(w_1, Sk_2)$ for Σ -solvability. This constraint is not Σ -solvable since $\Sigma_{2wm} \not\models \exists K_A(0, w_1) \wedge K_B(w_1, Sk_2)$. But it would be Σ -solvable if Σ contained the reflexivity axiom—e.g. if we were using the modal logic T .

The efficiency of a constraint deduction system depends critically upon the capability of the procedure to simplify constraints by replacing them with equivalent, but shorter constraints. This derivation uses one obvious constraint simplification. At step (20) a ground atom is deleted because it is Σ -solvable. Ideally, we would like to reduce the constraint to an equivalent maximally reduced [Buntine, 1988] or smallest constraint. In (21), the constraint $K_B(Sk_2, w_2) \wedge K_A(0, w_1) \wedge K_B(w_1, w_2)$ could be simplified to $K_B(Sk_2, w_2)$, but this is a non-trivial simplification. It requires realizing that for any ground terms t_1 and t' such that $\Sigma_{2wm} \models K_B(t_1, t')$, there will not be another t_2 such that $\Sigma_{2wm} \models K_B(t_2, t')$. In other words, if one interprets the minimal Herbrand model of Σ_{2wm} as a system of interconnected worlds, there can only be one distinct path to any world—the set of interconnected worlds form a tree.

This property is exploited by the specialised procedure, to be presented in the next section, for simplifying and testing the Σ -solvability of constraints that arise in a reified modal logic. The special procedure should also be incremental in that work done in testing for Σ -solvability at one point, does not

need to be redone later, but is rather stored in the representation of the constraints. Observe that the work done in the example above on testing for Σ -solvability at each step is redone again and again at later steps. If the testing for Σ -solvability is done by some sort of deduction (e.g. SLD resolution), the fact that the only term which can instantiate w_2 is $SK(Sk_3)$ (in step (21) above) will be deduced again and again—whenever the test for Σ -solvability is performed. The specialised constraint processing method, to be presented in the next section, is incremental.

Before continuing to the next section, we give an illustration (based on an example in [Moore, 1980]) of the operation of the rule of constrained paramodulation.

$$(23) \square num(bill) = num(mary)$$

$$(24) \exists z \square z = num(bill)$$

An agent (23) knows that Bill's phone number is the same as Mary's phone number. Additionally, (24) the agent knows Bill's phone number. The hypothesis to be proven is (25) that the agent knows Mary's phone number.

$$(25) \exists z \square z = num(mary)$$

The translation into the reified modal logic is as follows, where (28) is the translation of the negation of sentence (25):

$$(26) \forall w_1 : K(0, w_1)$$

$$num(bill, w_1) = num(mary, w_1)$$

$$(27) \exists z \forall w_1 : K(0, w_1) z = num(bill, w_1)$$

$$(28) \neg \exists z \forall w_1 : K(0, w_1) z = num(mary, w_1)$$

Putting (26)—(28) in clausal form results in the following:

$$(29) num(bill, w_1) = num(mary, w_1) / K(0, w_1)$$

$$(30) f = num(bill, w_1) / K(0, w_1)$$

$$(31) \neg(z = num(mary, sk(z))) / \emptyset$$

The constraint theory for this problem contains the clause $K(0, sk(z))$ along with the accessibility axioms for the modal logic being used. In this particular example, no accessibility axioms are needed to obtain a proof.

A proof that (29)—(31) are unsatisfiable is obtained by first paramodulating clause (29) into clause (30) to obtain (32).

$$(32) f = num(mary, w_1) / K(0, w_1)$$

Next, the empty clause (33) is obtained by resolving clauses (32) and (31).

$$(33) \square / K(0, sk(z))$$

Compare this with the situation where (23) is replaced by

$$(23') num(bill) = num(mary).$$

The clausal reified modal logic translation of (23') is

$$(29') num(bill, 0) = num(mary, 0) / \emptyset.$$

Now (29') can be paramodulated into (30) to yield

$$(32') f = num(mary, 0) / K(0, 0)$$

provided $K(0, 0)$ is Σ -solvable. Since it is not, the proof fails. The step would, however, succeed in a reflexive logic because it would then be the case that $\Sigma \models K(0, 0)$. But the proof of the unsatisfiability of (29'), (30) and (31) still fails because (32') and (31) can not resolve.

4 Testing for Σ -Solvability

This section presents an alternative representation for the constraints of a reified modal logic and a procedure for testing their Σ -solvability. The procedures are specialised unification algorithms that operate on constraints represented as sequences of symbols that range over or denote worlds. This method has been utilized in a number of the related methods [Ohlbach, 1988, Wallen, 1990, Jackson and Reichgelt, 1989, Auffray and Enjalbert, 1989] mentioned earlier; our contribution is to show that these methods can be viewed as algorithms for testing Σ -solvability of constraints. The method is applicable to modal logics for which the translation procedure adds some subset of the reflexive, symmetric and transitive axioms to the constraint theory (along with the serial axiom). It is not applicable to logics that call for the Euclidian axiom to be added to the constraint theory.²

Let us define a sequence as a unary tree with labeled edges and vertices. The root label is always the constant 0, the other vertices are labeled with terms and the edge labels are the K predicate symbols. The constraints that occur in the reified modal logic can be represented as sets of sequences. For example, $K(0, Sk_1) \wedge K(Sk_1, w_1) \wedge K(w_1, f(w_1)) \wedge K(f(w_1), w_2)$ is represented as $\{(0 \xrightarrow{K} Sk_1 \xrightarrow{K} w_1 \xrightarrow{K} f(w_1) \xrightarrow{K} w_2)\}$.

A constraint theory can also be represented as sequences. Recall that the definite clauses within Σ

²Of the 9 distinct modal logics to which the general framework is applicable, the special procedure for testing the Σ -solvability of constraints is not applicable to $KD5$ and $KD45$. It is applicable to $S5$ because, although $S5$ is a Euclidian logic, the conditions on the accessibility relation can be axiomatised without the Euclidian axiom.

have come from two sources—Skolemization and the accessibility axioms. All of the accessibility axioms are represented implicitly in the sequence unification procedure. The remaining definite clauses that make up Σ are represented as sequences. For example, the clause $K(0, w_1) \rightarrow K(w_1, sk(w_1))$ is represented as $(0 \xrightarrow{K} w_1 \xrightarrow{K} Sk(w_1))$. Now Σ is a set of sequences and each of the constraints attached to the clauses is a set of sequences.

It is an important property of the sequence representation of the constraints and Σ that for any symbol occurring in the sequences resulting from the translation of a set of modal sentences, there is only one distinct sequence terminating in that symbol. This is guaranteed by the following theorem.

Theorem 8 *The constraint theory Σ and all constraints on the A-clauses in a reified modal logic when represented as a set of sequences form a tree with uniquely labeled nodes—i.e. there is only one path from the root to any node in the tree.*

Therefore, for any symbol occurring in a set of sequences (representing either a constraint or Σ in a reified modal logic), there is, in the set of prefixes of the sequences in the set, a unique sequence terminating in that symbol.

Let us consider how the reified modal logic representation of the wise-man problem can be represented with the sequence notation. As there are two accessibility relations, two distinct labels (K_A and K_B) are needed for the edges of the sequences. Clauses (15) – (18) can be written with the constraints in sequence form as follows:

$$(34) \text{ (axiom)} \quad W'(A, w_1) \vee \neg W'(A, w_2) \\ / \{(0 \xrightarrow{K_A} w_1 \xrightarrow{K_B} w_2)\}$$

$$(35) \text{ (axiom)} \quad W'(A, w_2) \vee W'(B, w_2) \\ / \{(0 \xrightarrow{K_A} w_1 \xrightarrow{K_B} w_2)\}$$

$$(36) \text{ (axiom)} \quad \neg W'(B, Sk(w_1)) / \theta$$

$$(37) \text{ (axiom)} \quad \neg W'(A, Sk_2)) / \theta$$

The constraint theory Σ_{2w} is represented as a set of two sequences— $\{(0 \xrightarrow{K_A} Sk_2), (0 \xrightarrow{K_A} w_1 \xrightarrow{K_B} Sk(w_1))\}$.

Using this representation, the sequence unification algorithm, shown in Figure 3, can be used to determine the Σ -solvability of the constraints. The input to the algorithm consists of L_1 and L_2 (the two literals of opposite polarity from a step of potential constrained resolution of two clauses), the constraints C_1 and C_2 (both sets of sequences) from the clauses and a constraint theory Σ (a set of sequences). The output of the algorithm is either *FAIL*, indicating that L_1 and L_2 do not share any Σ -ground instances and therefore that the new clause cannot

Input: two expressions L_1 and L_2 , and three sets of sequences C_1 , C_2 , and Σ . L_1 and C_1 share no variables with L_2 and C_2 .

Output: *FAIL* or a set of sequences C and a substitution λ .

If L_1 and L_2 are not unifiable
halt and report *FAIL*

Else

$\theta := \text{mgu}(L_1, L_2)$;

$\lambda := \{\}$;

$C := C_1 \cup C_2$;

For each variable w_i such that $w_i\theta \neq w_i$, do the following:

$s :=$ the prefix ending in w_i of a sequence in C ;

If $w_i\theta$ is a variable then

$s' :=$ the prefix of a sequence in C $w_i\theta$

Else ($w_i\theta$ is of the form $f(t_1, \dots, t_n)$)

for some $n \geq 0$)

$s' :=$ the prefix of a sequence in Σ such that it ends with $f(t'_1, \dots, t'_n)$ for some $t'_1 \dots t'_n$;

If $\text{UNIFY}(s, s')$ succeeds with substitution γ then

$\lambda :=$ the composition of λ and γ

$C := C\gamma$

Else

halt and report *FAIL*;

If the procedure finishes without reporting *FAIL*, C is returned by the procedure along with λ .

Figure 3: Constraint Solving Procedure

be deduced, or a new constraint C (equivalent to $(C_1 \wedge C_2)\theta$) in the form of a set of sequences and, a substitution λ . If the procedure does not result in *FAIL*, the substitution λ is applied to the body of the new clause, and the constraint of the new clause is C . The procedure *UNIFY* will vary according to what modal logic is being used. For D , *UNIFY* is ordinary term unification. The case of other logics will be discussed later.

In establishing the correctness of this procedure for testing Σ -solvability, we consider the least Herbrand model of a Σ in a reified modal logic for the modal logic D . In this model the connectivity relations (e.g. K_A , K_B) denote tuples of ground terms from the Herbrand Universe of the clauses in Σ . The connectivity relations together form a graph in which the edges are labeled with the connectivity relations and the nodes are the ground terms.

Theorem 9 *The graph of the connectivity relations from the least Herbrand model of a Σ for the modal logic D is a tree with the root being the term 0.*

We reserve the word *path* for a list, beginning with 0, of vertices of the graph representation of the least Herbrand model of Σ while the *sequences* are in the language. Both are lists of symbols that begin with the constant 0. The paths contain only ground terms, while the sequences may contain variables as well. Since the graph of the connectivity relations forms a tree rooted at 0, there can only be one path to any term in the graph. This is a property of the graph representation of the least Herbrand model of any constraint theory Σ from a reified set of sentences of the modal logic D . We may think of these terms as worlds and the paths as accessibility relations in a Kripke structure.

The basic idea here is that since there can be only one path to any ground term in the minimal Herbrand model of Σ , a constraint which specifies that there must be two paths (ground sequences corresponding to two paths) to a term cannot be Σ -solvable. Unification determines whether or not there are ground sequences (corresponding to paths in the least Herbrand model) in the intersection of two sequences. If unification succeeds, the result will be an expression which represents the set of ground sequences in the intersection of the two original sequences.

Consider the sequence representation of Σ_{2wm} . The set contains just two sequences— $\langle 0 \xrightarrow{K_A} Sk_2 \rangle$ and $\langle 0 \xrightarrow{K_A} w_1 \xrightarrow{K_B} Sk(w_1) \rangle$. When reasoning over the minimal Herbrand model of Σ_{2wm} , the only ground terms (worlds) we know about are 0, Sk_2 and those terms with Sk as the outermost function symbol. The only connectivity relations we know about are those summarized by the sequences. There is only one path from 0 to any of the other ground terms from the Herbrand universe.

As an illustration of the deductive process with the sequence representation of constraints, consider the refutation for the two wise-man puzzle when represented with clauses (34) – (37) and the sequence representation of Σ_{2wm} . At each step, the sequence unification procedure is given a set of sequences from each parent. In this simple problem, the procedure *UNIFY* is only called once by each invocation of the sequence unification procedure.

- (38) (from 34,37) $\neg W'(A, w_3)$
 $\quad / \{ \langle 0 \xrightarrow{K_A} Sk_2 \xrightarrow{K_B} w_3 \rangle \}$
- (39) (from 35,38) $W'(B, w_2) / \{ \langle 0 \xrightarrow{K_A} Sk_2 \xrightarrow{K_B} w_2 \rangle \}$
- (40) (from 36,39) \square

In step (38) the primary deduction mechanism (resolution) has returned the substitution $\{SK_2/w_1\}$. Thus we must *UNIFY* the sequences $\langle 0 \xrightarrow{K_A} w_1 \rangle$ and $\langle 0 \xrightarrow{K_A} Sk_2 \rangle$. This succeeds with the substi-

tution $\{SK_2/w_1\}$. The variable w_2 in (38) has been renamed to w_3 to avoid a clash in the next step. In step (39), the outer deduction results in the substitution $\{w_2/w_3\}$. We must then *UNIFY* $\langle 0 \xrightarrow{K_A} SK_2 \xrightarrow{K_B} w_3 \rangle$ and $\langle 0 \xrightarrow{K_A} w_7 \xrightarrow{K_B} w_2 \rangle$. This results in the substitution $\{SK_2/w_7, w_2/w_3\}$. In the last step, the empty clause is produced with the substitution $\{SK(w_1)/w_2\}$. This requires *UNIFY* to operate on $\langle 0 \xrightarrow{K_A} SK_2 \xrightarrow{K_B} w_2 \rangle$ and $\langle 0 \xrightarrow{K_A} w_1 \xrightarrow{K_B} SK(w_1) \rangle$. The unification succeeds with the substitution $\{SK_2/w_1, SK(w_1)/w_2\}$. Note that clauses (35) and (37) can not resolve because the sequences $\langle 0 \xrightarrow{K_A} w_1 \xrightarrow{K_B} w_2 \rangle$ and $\langle 0 \xrightarrow{K_A} Sk_2 \rangle$ do not unify; they are of different lengths. Notice that the problems of increasingly large constraints and of work performed at earlier steps needing to be redone have been eliminated.

Since, in the case of the modal logic D , the procedure *UNIFY* is ordinary term unification, sequences which are not the same length cannot be made the same by any substitution and therefore cannot unify. Now consider the least Herbrand model of a Σ from the reified modal logics for a logic with additional conditions on the accessibility relation (e.g. T , S_4 , and S_5). Additional connections are created in the graph representation of the least Herbrand model of Σ by the axioms of reflexivity, symmetry, and transitivity. It is no longer the case that there is a unique path to every term. But in all these cases, the set of terms and paths to those terms may be represented as a tree which is then closed under the appropriate axioms. This closure is captured by performing unification in the presence of the additional axioms. It is no longer the case that the sequences must have the same length in order to unify. For example, in the presence of reflexivity, the clauses (35) and (37) can resolve because the sequences $\langle 0 \xrightarrow{K_A} w_1 \xrightarrow{K_B} w_2 \rangle$ and $\langle 0 \xrightarrow{K_A} Sk_2 \rangle$ do unify with substitution $\{Sk_2/w_1, Sk_2/w_2\}$.

The procedure *UNIFY* from Figure 3 is unification in the presence of the appropriate axioms. With the additional axioms, unification is no longer guaranteed to find a single maximally general unifier. It may produce multiple maximally general unifiers—but only a finite number. Consider the unification of sequences $\langle 0 \xrightarrow{K_A} w_1 \xrightarrow{K_A} Sk \rangle$ and $\langle 0 \xrightarrow{K_A} w_2 \rangle$. Unification in the presence of reflexivity yields two maximally general unifiers— $\{0/w_1, Sk/w_2\}$ and $\{Sk/w_1, Sk/w_2\}$. Now that *UNIFY* returns a set of substitutions, the sequence unification procedure illustrated in Figure 3 must be modified to return a set of pairs, with each pair consisting of a substitution and a set of sequences. The production of multiple maximally general unifiers is a result of this particular representation for the constraints and is not a characteristic of the general method. In the

case of (single modality) S5, the number of unifiers can be limited to 1 by exploiting the fact that the worlds form equivalence classes and every world is connected to every other world in its equivalence class.

The correctness of the sequence unification procedure as a method for testing Σ -solvability of constraints is established by the following theorem.

Theorem 10 *Given a variable disjoint L_1 , C_1 and L_2 , C_2 and Σ , the sequence unification procedure succeeds if and only if L_1 and L_2 have a most general unifier θ and the constraint $C = (C'_1 \wedge C'_2)\theta$ is Σ -solvable—i.e. $\Sigma \models \exists (C_1 \wedge C_2)\theta$, where C'_1 and C'_2 are the K-atom representations of the sets of sequences C_1 and C_2 . When successful, the final result of this procedure is the sequence representation of a constraint equivalent to $(C'_1 \wedge C'_2)\theta$.*

The sequence unification procedures found in [Wallen, 1990, Ohlbach, 1988, Jackson and Reichgelt, 1989, Auffray and Enjalbert, 1989] can be shown to be variants of the procedure laid out above.

5 Varying Domain Logics

In constant domain modal logics, the same entities exist at all possible worlds. Both the Barcan formula (41) and the converse Barcan formula (42) are theorems.

$$(41) \forall z \square A(z) \rightarrow \square \forall z A(z)$$

$$(42) \square \forall z A(z) \rightarrow \forall z \square z A(z)$$

In the varying domain case neither formula may be a theorem or either one of the two (but not both) may be a theorem.

In translating varying domain modal logics into a reified modal logic, it is necessary to introduce a binary predicate E as a constraint atom. For example $\forall z P(z)$ is translated into $\forall z : E(z, 0) P(z, 0)$ which can be read as saying for all entities z that exist in the initial world, $p(z)$ is true in the initial world. In general, the atom $E(z, w_i)$ restricts the variable z to range over elements that exist in world w_i . The quantifier rules in the translation function T_i are replaced with the following:

- $T_i(\forall z \varphi) = \forall z' : E(z', w_i) T_i(\varphi)$
- $T_i(\exists z \varphi) = \exists z' : E(z', w_i) T_i(\varphi)$

It is also necessary to add $\forall w \exists z E(z, w)$ to the constraint theory to ensure that the domains in each world are non-empty. If the Barcan formula is to be a theorem, it is necessary to add

$$(43) \forall w_i, w_j (K(w_i, w_j) \wedge \forall z E(z, w_j)) \rightarrow E(z, w_i)$$

to the constraint theory. Likewise, if the converse Barcan formula is to be a theorem of the logic,

$$(44) \forall w_i, w_j (K(w_i, w_j) \wedge \forall z E(z, w_i)) \rightarrow E(z, w_j)$$

needs to be added.

In the varying domain case both K and E are constraint atoms. The transformation to Skolem normal form and the results of Section 3 are general enough to apply when there are more than one constraint atoms.

Consider a varying domain logic with no additional axioms. Let's try to prove (42), the converse Barcan formula. The translation of (42) is:

$$(45) \exists w_1 : K(0, w_1) \exists z : E(z, w_1) \neg A(z, w_1) \vee \\ \forall z : E(z, w_1) \forall w_1 : K(0, w_1) A(z, w_1)$$

When the formula is negated and placed in clausal form, the result is two clauses as follows:

$$(46) A(z, w_1) / K(0, w_1) \wedge E(z, w_1)$$

$$(47) \neg A(f, sk) / \emptyset$$

The constraint theory for this problem contains $E(f, 0)$ and $K(0, sk)$.

Note that there is a potential resolvent of (46) and (47). But it is then necessary to show that $\Sigma \models E(f, sk)$. Since $\Sigma \not\models E(f, sk)$, this step of resolution is not possible and the proof of (42) fails. But if (44) were added to Σ , (46) and (47) could resolve to yield (48).

$$(48) \square / K(0, sk) \wedge E(f, sk)$$

Now it is the case that $\Sigma \models K(0, sk) \wedge E(f, sk)$ and a proof of (42) is obtained.

The constraint solving procedure of Section 4 can be extended to the varying domain case following the method utilized by Wallen[1990]. The basic idea here is that the sequence representation of the constraints is augmented with a table that associates each term that occurs in the first argument position of the E atoms (in the K and E -atom representation of the constraints) with the sequence terminating in the second argument position of the E atom. In the case of varying domain logics (without (43) and (44) in the K and E -atom representation of Σ), the substitution of term t_1 for t_2 requires a check to see if $UNIFY(s(t_1), s(t_2))$ succeeds where $s(t)$ is the sequence associated with the term t . If for example the converse Barcan formula is to be a theorem of the logic (i.e. (44) is in the K and E -atom representation of Σ), it is necessary to verify that either $UNIFY(s(t_1), s(t_2))$ or $UNIFY(s'(t_1), s(t_2))$ succeeds where $s'(t_1)$ is some subsequence of $s(t_1)$.

6 Conclusion

A general framework for constructing and proving the correctness of methods for automated deduction in modal logic has been discussed and illustrated. It provides a systematic method for converting a non-modal deduction procedure and its proof of correctness to a corresponding modal deduction procedure with a proof of correctness. The framework achieves significant generality in that it provides for the extension of a wide range of non-modal systems to corresponding modal systems and that this can be done for a wide range of modal logics. The ease with which we were able to add equality to the framework speaks for its usefulness.

Most of the features of the members of the general class of related methods [Wallen, 1990, Ohlbach, 1988, Jackson and Reichgelt, 1989, Auffray and Enjalbert, 1989] can be recast as particular instantiations of our general framework. The approach developed here has the advantage of producing relatively simple proofs of soundness and completeness. They follow from general results on substitutional reasoning. A deeper understanding of how these related approaches work has been obtained. Also, the general framework has provided a common set of principles which relate various modal and non-modal deduction methods.

Many of the related methods are limited to serial modal logics. Ohlbach adds an additional inference rule in order to handle the non-serial case. Wallen, using a version of the matrix method which does not require prenexing, is able to handle non-serial logics without any additional machinery. Our framework explains why non-seriality is a problem. The framework specifies when a simple modification of an ordinary inference rule that works on prenex Skolem normal form formulas suffices to maintain completeness in the constrained case. Non-serial logics fall outside the boundary as the constrained sentences cannot be put in equivalent prenex form.

On the other hand, logics for which it is necessary to add the Euclidian axiom to Σ are not considered by the other researchers. They fit readily into the overall framework, but the difficulty lies in extending the method of sequence unification to a test for Σ -solvability when Σ contains the Euclidian axiom.

The approach has led to a number of problems, which are all topics of our current and future research. These include finding a representation for constraints that avoids generating multiple maximally general unifiers, developing a procedure for testing for Σ -solvability in the presence of the Euclidian axiom, extending the framework to cover inference rules that work on non-prenex formulas, and also extending the method to a wider class of modal

and intensional logics.

References

- [Auffray and Enjalbert, 1989] Yves Auffray and Patrice Enjalbert. Modal theorem proving: An equational viewpoint. In *IJCAI-89*, pages 441–445, Detroit, August 1989.
- [Buntine, 1988] Wray Buntine. Generalised subsumption and its applications to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, September 1988.
- [Fitting, 1983] Melvin C. Fitting. *Proof Methods for Modal and Intuitionist Logics*. D. Reidel Publishing Co., Dordrecht, 1983.
- [Frisch, 1991] Alan M. Frisch. The substitutional framework for sorted deduction: Fundamental results on hybrid reasoning. *Artificial Intelligence*, 1991. (to appear, an earlier version appeared in KR89).
- [Genesereth and Nilsson, 1987] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufman, Los Altos, California, 1987.
- [Jackson and Reichgelt, 1989] Peter Jackson and Han Reichgelt. A general proof method for modal predicate logic. In *Logic-Based Knowledge Representation*, chapter 8, pages 177–228. MIT Press, Cambridge, Massachusetts, 1989.
- [Jaffar and Lassez, 1987] Joxan Jaffar and Jean-Louis Lassez. Constraint logic programming. In *POPL-87*, pages 111–119, Munich, January 1987.
- [Kripke, 1963] S. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.
- [Moore, 1980] Robert Moore. Reasoning about knowledge and action. Technical Note 191, SRI International, October 1980.
- [Morgan, 1976] C.G. Morgan. Methods for automated theorem proving in nonclassical logics. *IEEE Transactions on Computers*, C-25:852–862, 1976.
- [Ohlbach, 1988] Hans Jürgen Ohlbach. A resolution calculus for modal logic. In E. Lusk and R. Overbeek, editors, *CADE-9*, volume 310 of *Lecture Notes in Computer Science*, pages 500–516. Springer-Verlag, Berlin, May 1988.
- [Van Hentenryk, 1989] Pascal Van Hentenryk. *Constraint Satisfaction in Logic Programming*. MIT Press, Cambridge, MA, 1989.
- [Wallen, 1990] Lincoln A. Wallen. *Automated Proof Search in Non-Classical Logics: Efficient Matrix Proof Methods for Modal and Intuitionistic Logics*. The MIT Press, Cambridge, Massachusetts, 1990.

Credulous vs. Sceptical Semantics for Ordered Logic Programs

D. Gabbay

Dept. of Computing
Imperial College
180 Queen's Gate
London SW72BZ, UK

E. Laenens

Dept. of Math. and Computer Science
University of Antwerp, UIA
Universiteitsplein 1
B2610 Wilrijk, Belgium

D. Vermeir

Dept. of Math. and Computer Science
University of Antwerp, UIA
Universiteitsplein 1
B2610 Wilrijk, Belgium

Abstract

We present a semantic approach to the characterization of credulous and sceptical reasoning mechanisms, within the framework of ordered logic. One of the advantages of our approach is that it integrates tightly with "conservative" ordered logic semantics which is known to generalize "classical" (stable and well-founded) logic programming semantics. This allows us to compare the conservative and credulous approaches, thus providing insight in the fundamental properties of both reasoning paradigms. It turns out that maximal credulous models are extensions of conservative stable models while the (unique) minimal credulous model, called the sceptical model, is a restriction of the well-founded model.

1. MOTIVATION

Consider a logic program $C_1 = \{\neg q \rightarrow p\}$. The computation of this program assumes that, since q is not a head of any clause, $\neg q$ is part of the data. Suppose we relinquish this principle and adopt the principle of asking an *adviser* what to do with $\neg q$. The adviser might say that $\neg q$ succeeds or might say that $\neg q$ fails. The adviser might have his own program to consult. If his program is C_2 , he might run the goal q (or $\neg q$), look at what he gets and then advise. To make the situation symmetrical and general we must allow for Horn programs to have rules with both q and $\neg q$ (i.e. literals) in heads and bodies and have any number of negotiating advisers. Thus we can have $C_2 = \{\neg q\}$, $C_1 = \{\neg q \rightarrow p\}$ and C_1 depends on C_2 . Ordered logic^{Lae90a, Lae90b, Ver89a} develops and studies various aspects of such an advisor system which is modeled as a partially ordered set of theories. Such a logic is useful, e.g. for multi-expert systems where we want to represent the knowledge of several experts in a single system. Experts may then be ordered according to an "advisory" or a relative preference relation.

A problem to consider is what happens when we have several advisers that are in conflict. For example, C_1 depends on C_2 and C_1 depends on C_3 . The two

advisers, C_2 and C_3 , may be in conflict. One may advise $\neg q$, the other q . How to decide? There are several options: in a *conservative* approach, one would not make an overall conclusion re. q , while a *credulous* approach could yield two models: one with q , the other with $\neg q$. In this work, we present a semantic approach to the characterization of credulous and sceptical reasoning mechanisms, within the framework of ordered logic^{Lae90a}. One of the advantages of our approach is that it integrates tightly with "conservative" ordered logic semantics which is known^{Lae90b} to generalize "classical" (stable and well-founded) logic programming semantics. This allows us to compare the conservative and credulous approaches, thus providing insight in the fundamental properties of both reasoning paradigms. It turns out that our formalization of credulous reasoning extends the well-founded-stable hierarchy of (partial) models in that stable models can be extended to credulous models while the (unique) well-founded model is an extension of the (unique) minimal credulous model, which is called the sceptical model. The present work also sheds light on the underlying mechanisms that may cause a theory to have multiple "best" models: in the conservative approach, multiplicity (of stable models) is caused by making "choices" which are not based on "assumptions", while the credulous approach allows further extensions of a (stable) model by making (conservative) assumptions.

We start off by presenting a summary of the relevant definitions and results on ordered logic.

The proofs of the results presented in this paper can be found in Chapter 8 of^{Lae90c}.

2. ORDERED LOGIC PROGRAMS

In this section, we introduce the basic concepts and notations of ordered logic. For a more detailed description with examples, we refer to^{Lae90b}

The basic tokens of the language are terms, formulas and literals where a *term* is defined as a variable or a

constant Llo87a. An *atomic formula* or *atom* of the language is of the form $p(t)$, where p is a predicate symbol with arity n ($n \geq 0$) and t is a sequence of n terms (*arguments of the atom*). A *literal* is either an atom (*positive literal*) or its negation (*negative literal*). A term, predicate or literal is *ground* if it is variable-free.

Two literals are complementary if they are of the form A and $\neg A$, for some atom A . In general, given a literal A and a set of ground literals X , $\neg A$ denotes the complement of A and $\neg X$ denotes the set of literals $\{\neg B \mid B \in X\}$. Moreover, $\text{pos}(X)$ (resp. $\text{neg}(X)$) denotes the set of all positive (resp. negative) literals in X .

A *negative rule* (or simply, a *rule*) is a formula of the language represented as follows:

$$Q_1, \dots, Q_m \rightarrow Q_0$$

where Q_0, \dots, Q_m are literals, Q_0 is the *head* of the rule, and Q_1, \dots, Q_m is the *body* of the rule. If Q_0 is positive then the rule is a *seminegative rule*; moreover, if also Q_1, \dots, Q_m are all positive then the rule is a *positive rule* (or Horn clause). Given a rule r , $H(r)$ denotes the head of r and $B(r)$ denotes the set of all literals in the body of r . A rule is *ground* if it is variable-free. A ground rule is a *fact* if it has an empty body.

A *negative program* is a set of rules. If all rules are seminegative (resp. positive) then the program is called a *seminegative program* (resp. *positive program*).

Let P be a negative program. The Herbrand Universe of P (denoted by H_P) is the set of all possible ground terms. The Herbrand Base of P (denoted by B_P) is the set of all possible ground atoms whose predicate symbols occur in P and whose arguments are elements of H_P . A *ground instance* of a rule r in P is a rule obtained from r by replacing every variable X in r by $\phi(X)$, where ϕ is a mapping from the set of all variables occurring in P to H_P . The set of all ground instances of all rules in P is denoted by *ground*(P). Any subset of $B_P \cup \neg B_P$ is called an *interpretation* for P . We say that an interpretation X is *consistent* if there are no two literals A and B in X such that $A = \neg B$. Finally, we denote by X the set $\{p \in B_P \cup \neg B_P \mid \text{neither } p \text{ nor } \neg p \text{ is in } X\}$ and by \bar{X} the set $(B_P \cup \neg B_P) \setminus X$.

We now formally define the notion of ordered logic program.

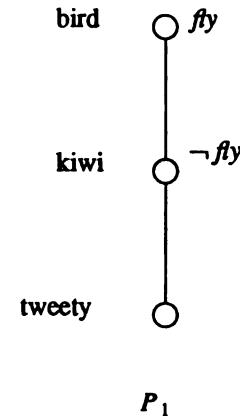
Definition 1.

An *ordered logic program* is a finite partially-ordered set of negative programs, called *components*, where " \leq " is the partial order. \square

The " $<$ " relation is a sort of isa hierarchy for the components and provides the ground for inheritance.

(As usual, " $<$ " denotes the restriction of " \leq " to all pairs of distinct components.)

Throughout all examples of this work, an ordered logic program P is represented as a pair $\langle C, L \rangle$ where C is the set of components and L is the relation " $<$ ", or as a directed acyclic graph (dag) in which the nodes represent the components and the arcs the relation " $<$ ". As an example consider the ordered logic version P_1 of the well known "tweety" example: P_1 has components $\text{bird} = \{\text{fly}\}$, $\text{kiwi} = \{\neg \text{fly}\}$ and $\text{tweety} = \emptyset$ ordered by $\text{tweety} < \text{kiwi} < \text{bird}$.



P_1

It turns out that an ordered logic program has several meanings, one for each of its components. Moreover, the knowledge defined at a component, i.e. the local knowledge, does not constitute the entire knowledge about that component as a component inherits information from other components. Therefore, the meaning of a particular component in an ordered logic program is defined by all rules that are visible to this component, i.e. all local rules (defined in this component) as well as all global rules defined in components that are higher up in the component-hierarchy induced by the partial order.

Definition 2.

Given an ordered logic program P and a component C of P . An *interpretation* for P in C is any interpretation of C^* , where C^* denotes the negative program $\{r \mid r \in C' \text{ and } C \leq C'\}$. \square

In the next section, we briefly discuss the conservative semantics for ordered logic.

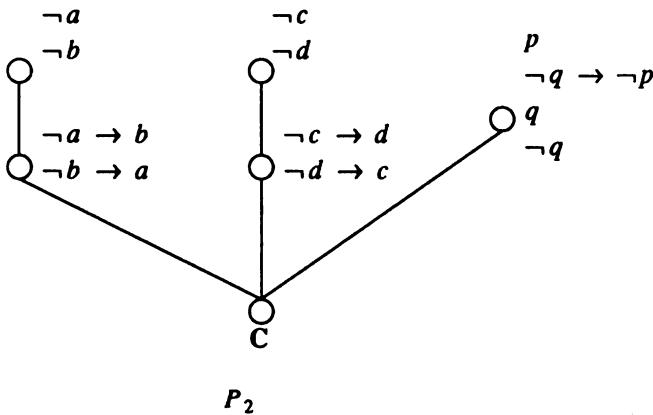
3. PARTIAL MODEL SEMANTICS

For a detailed discussion of the well-founded and stable partial model semantics, we refer to Lae90c. As was explained in the previous section, an ordered logic program has several meanings, one for each component. In a given component, its semantics depends on the rules that are visible to it, i.e. all local rules (defined in this component) as well as all global rules defined in components that are higher up in the

component-hierarchy induced by the partial order. Hence, an interpretation reflects the meaning of a component if it satisfies all rules that are visible to this component. In other words, for a consistent interpretation M to be a model for a component, it suffices that it satisfies all local and global rules of this component. Classically, this would amount to demanding that each visible rule r in $\text{ground}(C^*)$ is either non-applicable in M , i.e. $B(r) \not\subseteq M$, or applied in M , i.e. $B(r) \subseteq M$ and $H(r) \in M$. In the case of an ordered logic program, this is not enough^{Lae90d, Lae90a} as it cannot deal with a situation where several competing rules (i.e. rules with complementary heads p and $\neg p$) are applicable. So we add a third possibility, saying that a rule is also satisfied if it is defeated by a competitor, i.e. it has an applicable competitor where a competitor is a visible rule with complementary head that is defined in a component that is not strictly higher in the component hierarchy. Let $c(r)$ denote the component in which the rule r is defined. For a given ordered logic program P and a component of it C , the set of competitors of a rule r in $\text{ground}(C^*)$ is

$$\text{comp}_{P|C}(r) = \{\hat{r} \in \text{ground}(C^*) \mid H(\hat{r}) = \neg H(r) \text{ and } c(\hat{r}) \not\succ c(r)\}$$

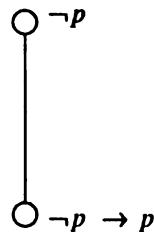
r is defeated in M by a rule \hat{r} in $\text{comp}_{P|C}(r)$ if \hat{r} is applicable in M . Such a competitor will be called a defeater of r . Summarizing, an interpretation M is a model for an ordered logic program in a component of it if each rule that is visible from this component is either non-applicable or applied or defeated. As an example, the interpretation $\{\neg \text{fly}\}$ is a model of P_1 since the rule fly at bird is defeated by the applied rule $\neg \text{fly}$ at kiwi . As another example, consider P_2 .



Both $\{\neg a, b, \neg c, d, p\}$ and $\{a, \neg b, c, \neg d, p\}$ are models of P_2 , as can easily be verified.

Those interpretations that are not models can be thought of as interpretations that are 'too poor', in the sense that they are not deductively closed. Some ordered logic programs such as P_3 have no models: Programs like P_3 are called "contradictory" in^{Lae89a}.

In^{Lae91a} the weaker notion of "partial model" was introduced in order to be able to give a meaningful



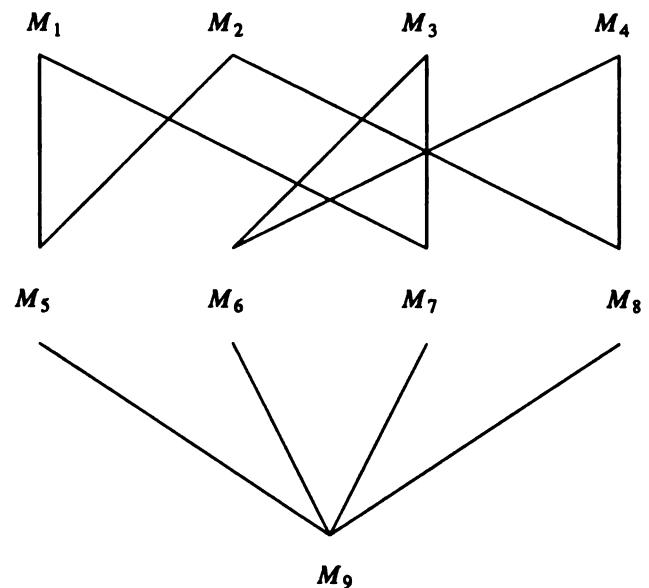
P_3

interpretation to any ordered logic program. Intuitively, the notion of partial model is obtained by relaxing the definition of defeater: instead of demanding that a rule's competitor be applicable, we will require that its body not be "false", i.e. it is either true (in which case the competitor is applicable) or undefined^{Ben00a, Prz89a, You90a}. Reconsidering the example program P_3 , the empty set is a partial model as the literal $\neg p$ is undefined and therefore the (non-applicable) rule $\neg p \rightarrow p$ acts as a defeater of $\neg p$.

In order to formalize this intuition, we need to characterize what it means for a literal to be "false" w.r.t. a given interpretation. We cannot adopt the traditional logic programming definition saying that a ground literal is false w.r.t. a given interpretation if its negation is true w.r.t. this interpretation, since this traditional definition of falsity makes only sense in the context of some closed world assumption where the negation of each literal that is considered false w.r.t. the interpretation is added to this interpretation. In other words, a literal is not false because its negation is in the interpretation, but the negation of a literal is in the interpretation because the literal is false. An excellent illustration of this is the fixpoint computation of the well-founded model in^{Gel88a} where each iteration adds besides the positive literals as inferred using the immediate consequence transformation, also the negation of each literal in the unfounded set which is exactly the set of ground literals that are already known to be false. Since ordered logic makes no use of a closed world assumption, it follows that we need to know the ordered-logic equivalent of the unfounded set in order to obtain a correct notion of falsity. To this end, we generalize the concept of unfounded set as defined in^{Gel88a} to ordered logic programs. As in the classical case, the greatest unfounded set w.r.t. an interpretation is the set of literals that are false w.r.t. this interpretation. Intuitively, a rule whose body is false, i.e. it contains unfounded literals, is out of order and should not be accepted as a defeater of some other rules; only rules of which the body is either false or undefined can act as defeaters. We say that a rule r in $\text{ground}(C^*)$ is **defeasible** if it has a competitor \hat{r} in $\text{comp}_{P|C}(r)$ whose body is not false, i.e. the intersection of $B(\hat{r})$ with the greatest unfounded set is empty.

Hence the following definition of partial model, which weakens the definition of model. An interpretation M is a **partial model** for an ordered logic program in a component of it if each rule $C \rightarrow p$ that is visible from this component is either non-applicable or applied or defeasible. Intuitively, a partial model M guarantees that there are no 'pending' rules, i.e. rules that definitely should, and are not applied in M . In other words, those interpretations that are not partial models can be thought of as interpretations that are 'too poor', in the sense that they are not deductively closed. On the other hand however, partial models can still be 'too rich' in that they may contain literals that cannot be actually inferred. We call these literals, that are not justified by an applicable non-defeated and non-defeasible rule, assumptions. An assumption set of a partial model M is a subset X of this partial model such that for each literal p of X , each rule r in $\text{ground}(C^*)$ with $H(r)=p$ is either non-applicable in M or defeated in M or defeasible in M or contains assumptions ($B(r) \cap X \neq \emptyset$). The last possibility avoids circularities where a literal l_1 motivates another literal l_2 and vice versa. It is easily verified that assumption sets are closed under union, so we can talk about the greatest assumption set of a partial model M . It is evident that our interest goes to **assumption-free partial models**, i.e. partial models that have an empty greatest assumption set, as they are fully inferrable as well as deductively closed, thus representing the exact meaning of the ordered logic program. It can be shown that each ordered logic program has a partial model. In particular, the least fixpoint $W_P|_C(\emptyset)$ of the (monotone) *ordered immediate consequence transformation*, $W_P|_C$, defined as $W_P|_C(I) = \{p \mid \text{there exists a rule } C \rightarrow p \text{ which is applicable but not defeasible in } I\}$, is an assumption-free partial model.

The set of all assumption-free partial models of an ordered logic program P form a partial order under the subset relation. The maximal elements of this partial order are called *stable partial models* while the minimal elements are called *well-founded partial models*. As an example, the partial-model hierarchy corresponding to the ordered logic program P_2 is depicted in below. Note that there are four stable partial models, $M_1 \dots M_4$, and one well-founded partial model $M_9 = \{p\}$, where $M_5 = \{a, \neg b, p\}$, $M_6 = \{\neg a, b, p\}$, $M_7 = \{c, \neg d, p\}$, $M_8 = \{\neg c, d, p\}$, $M_1 = M_5 \cup M_7$, $M_2 = M_5 \cup M_8$, $M_3 = M_6 \cup M_7$, and $M_4 = M_6 \cup M_8$. In ^{Lae91a} it was shown that there is but a unique well-founded partial model, which is equal to $W_P|_C(\emptyset)$. Another interesting result is that the intersection of all stable partial models is exactly the well-founded partial model in all cases but a very special type of (contradictory) ordered logic programs. Together with the proof theory (procedural semantics) for ordered logic developed in ^{Ver89b}, this gives a soundness and completeness result for ordered logic. For the relationship between partial models and "full" models, we refer to



^{Lae91a}

4. CREDULOUS AND SCEPTICAL MODELS

It was discussed before that the need for defeating arises where the classical approach fails, namely when it comes to the satisfaction of applicable competing rules. From this, it follows that an actual defeater, i.e. a rule that causes a rule to be defeated (not just defeasible), should definitely be applicable. The question then arises as to whether a defeater should be applied or not? If in an ordered logic program, every two competing rules are such that one is strictly preferred to the other, then the defeater of an applicable, non-applied rule is automatically applied. So, the answer only depends on whether we want to ignore equivalent (i.e. incomparable in the partial order) conflicting information or whether we want to make a choice in the case of equivalent conflicting information. It turns out that there are two alternative approaches, one taking the view that a defeater is an applicable competitor thus unifying conflicting information with no information, and the other taking the view that a defeater should be an applied competitor thus enforcing a choice. The former *conservative* approach was discussed in the previous section. Our objective now is to develop semantics for the second *credulous* approach and to discuss its relationship with the conservative approach.

Only a few changes to the definitions of the previous section are required: specifically, the new notion of defeating will be reflected in the credulous versions of (the definitions of) unfounded and assumption sets. Surprisingly, it will turn out that the new credulous semantics not only provides us with a more credulous interpretation of ordered logic programs, but also with an extremely sceptical interpretation.

Definition 3.

Let P be an ordered logic program and C a component of it. Given an interpretation I for P in C , a rule r in $\text{ground}(C^*)$ is **c-defeated** in I if there exists a rule \hat{r} in $\text{comp}_{P|C}(r)$ such that $B(\hat{r}) \cup H(\hat{r}) \subseteq I$. \square

Definition 4.

Let P be an ordered logic program, C a component of it, and I an interpretation for P in C . A subset X of $B_P \cup \neg B_P$ is a **c-unfounded set** of P in C w.r.t. I if for each p in X , every rule r in $\text{ground}(C^*)$ with $H(r)=p$ is either c-defeated in I or $B(r) \cap X \neq \emptyset$.

The **greatest c-unfounded set** of P in C w.r.t. I , denoted $U_{P|C}^c(I)$, is the union of all sets that are c-unfounded sets of P in C w.r.t. I . $U_{P|C}^c(I)$ is easily seen to be a c-unfounded set. \square

Lemma 1.

Let P be an ordered logic program, C a component of it, and I and J interpretations for P in C . If $I \subseteq J$, then $U_{P|C}^c(I) \subseteq U_{P|C}^c(J)$. \blacksquare

Clearly, $U_{P|C}^c(I)$ is a subset of $U_{P|C}(I)$ which fits our intuition that, in the current case, more literals should be true, i.e. more literals should be in I , due to the enforced choice when conflicts arise. From this observation, it follows immediately that, in general, we can expect more c-partial models than partial models as the number of defeasible rules increases (i.e. there are more c-defeasible rules than defeasible rules).

Definition 5.

Let P be an ordered logic program and C a component of it. Given an interpretation I for P in C , a rule r in $\text{ground}(C^*)$ is **c-defeasible** in I if there exists a rule \hat{r} in $\text{comp}_{P|C}(r)$ such that $(B(\hat{r}) \cup H(\hat{r})) \cap U_{P|C}^c(I) = \emptyset$. \square

Definition 6.

Let P be an ordered logic program and C a component of it. A consistent interpretation M is a **c-partial model** for P in C if every rule r in $\text{ground}(C^*)$ is either applied in M or non-applicable in M or c-defeasible in M . \square

The next definition is simpler than the conservative definition of assumption set. This is due to the fact that an applied rule can never be c-defeated w.r.t. an interpretation.

Definition 7.

Let P be an ordered logic program, C a component of it, and I an interpretation for P in C . A subset X of I is a **c-assumption set** of P in C w.r.t. I if for each p in X , every rule r in $\text{ground}(C^*)$ with $H(r)=p$ satisfies one of the following conditions:

- (a) r is non-applicable in I , or
- (b) r is c-defeasible in I , or
- (c) $B(r) \cap X \neq \emptyset$.

The **greatest c-assumption set** of P in C w.r.t. I , denoted $A_{P|C}^c(I)$, is the union of all sets that are c-assumption sets of P in C w.r.t. I . $A_{P|C}^c(I)$ is easily seen to be a c-assumption set. A member of $A_{P|C}^c(I)$ is called an **c-assumption**. I is said to be **c-assumption free** iff $A_{P|C}^c(I)=\emptyset$. \square

Interpretations that are free from c-assumptions are also free from c-unfounded literals.

Lemma 2.

Let P be an ordered logic program, C a component of it, and I an interpretation for P in C . Then $U_{P|C}^c(I) \cap I \subseteq A_{P|C}^c(I)$. \blacksquare

As for the conservative approach, we can show that the greatest c-assumption set is that part of the interpretation which is non-inferable. To this end, we define the **credulous enabled version** which corresponds to the inferable part of a c-partial model.

Definition 8.

Let P be an ordered logic program, C a component of P and M a c-partial model for P in C . The **credulous enabled version** of $\text{ground}(C^*)$ w.r.t. M , denoted by C_M^c , is the program containing all rules of $\text{ground}(C^*)$ that are applied in M and not c-defeasible in M . \square

Let us now apply the **immediate consequence transformation** T as defined for seminegative and positive programs to C_M^c . Given any interpretation I for P in C , $T_{C_M^c}(I) = \{A \mid \text{there exists a rule } r \text{ in } C_M^c \text{ such that } H(r)=A \text{ and } B(r) \subseteq I\}$.

Lemma 3.

Let P be an ordered logic program, C a component of P and M a c-partial model for P in C . Then $T_{C_M^c}$ is monotone and has a least fixpoint, denoted $T_{C_M^c}^*(\emptyset)$. Moreover, $T_{C_M^c}^*(\emptyset) \subseteq M$. \blacksquare

The members of $M - T_{C_M^c}^*(\emptyset)$ cannot be inferred in C_M^c : they are assumptions of M .

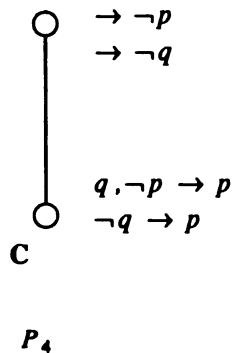
Theorem 1.

Let P be an ordered logic program and C a component of it. A c-partial model M for P in C is c-assumption free if and only if $T_{C^*}^*(\emptyset) = M$. ■

While the condition of c-defeating would be meaningless in the definition of c-assumption set, we next illustrate that the condition of c-defeasible is essential.

Example 1.

Consider the program P_4 depicted below, together with the interpretation $M = \{\neg p\}$.



Clearly, $U_{P_4|C}(M) = \emptyset$, and therefore, the credulous enabled version is also empty as each rule in P_4 is either c-defeasible in M or non-applicable in M . As expected (Theorem 1), we find that M is not assumption free because the rule $\rightarrow \neg p$ is c-defeasible in M . So, condition (b) in the definition of c-assumption set is essential. □

We next prove that a c-assumption-free c-partial model exists for any ordered logic program in any of its components, by means of the following transformation.

Definition 9.

Let P be an ordered logic program and C one of its components. Let I be the family of all interpretations of P in C . The credulous ordered immediate consequence transformation for P in C is the function $S_{P|C}: I \rightarrow I$ defined as follows: given an interpretation I , $S_{P|C}(I) = \{p \mid \text{there exists a rule } r \text{ in ground}(C^*) \text{ such that } H(r) = p, r \text{ is applicable in } I \text{ and not c-defeasible in } I\}$. □

Lemma 4.

For a given ordered logic program P and a component of it C , the transformation $S_{P|C}$ is monotone and has a least fixpoint. ■

We denote the least fixpoint of $S_{P|C}$ by $S_{P|C}^*(\emptyset)$.

Lemma 5.

Given an ordered logic program P and a component C of P . $S_{P|C}^*(\emptyset)$ is a c-assumption-free c-partial model for P in C . ■

We now consider the collection of all c-assumption-free c-partial models of an ordered logic program P in a component C , which is partially ordered by the set-inclusion relation.

Definition 10.

Let P be an ordered logic program, C a component of it and $\langle \mu, \subseteq \rangle$ the partially ordered set where μ denotes the set of c-assumption-free c-partial models for P in C . A c-partial model M for P in C is called

- **credulous** if M is maximal in $\langle \mu, \subseteq \rangle$,
- **sceptical** if M is minimal in $\langle \mu, \subseteq \rangle$. □

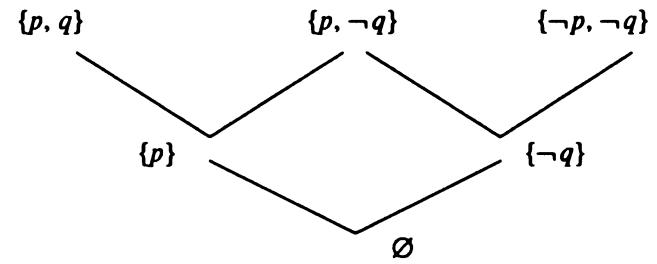
Example 2.

Consider the ordered logic program P_5 , which has but one component C .

$$C =$$

$$\begin{cases} & \\ & \rightarrow p \\ & \neg q \rightarrow \neg p \\ & \rightarrow q \\ & \rightarrow \neg q \\ \end{cases}$$

The partial order of c-assumption-free c-partial models for P_5 in C is depicted in the next figure. There are 3 credulous c-partial models and there is a single sceptical one.



□

While the previous example illustrates that an ordered logic program may have several alternative credulous c-partial models in a component, we can show that the uniqueness of the sceptical c-partial model is guaranteed.

Theorem 2.

Let P be an ordered logic program and C a component of it. Then $S_{P|C}^*(\emptyset)$ is the unique sceptical c-partial model. ■

5. SCEPTICAL PROOF THEORY

Our next goal is to develop a definition of "proof for properties of objects": for an ordered logic program P and a component C of it, we want a proof procedure that decides whether a given ground literal is true in C of P . In a monotonic environment, producing evidence for the truth of all literals in the body of a rule suffices to conclude the truth of the literal in the head of this rule. Obviously, our proof mechanism will be more complicated as it is essential to take into account the behavior of the competitors of this rule. Therefore, the proof procedure will have to show that any competitor of this rule is in fact "blocked", i.e. at least one literal of its body cannot become true. This is done by inferring not only positive conclusions like " p holds" but also negative ones like "demonstrably, p does not hold".

Proofs of properties of objects are conveniently represented as trees. The intuition is as follows. Let P be an ordered logic program and C a component of it. The nodes of a sceptical prooftree in C of P are labeled by adorned literals, the adornment being either "+" or "-". If a node n in a sceptical prooftree T has a positive adornment, e.g. p^+ , then the subtree of T with root n is again a sceptical prooftree which represents a proof for " p is true in C of P ". The proof is based on the idea that p must be true in C of P only if there is an applicable, not c-defeasible rule whose head is p , which means that (1) its body can be proven to be true and that (2) the body or the head of each competitor can be proven not to be true. If a node n in a sceptical prooftree has a negative adornment, e.g. p^- then the subtree with root n is again a sceptical prooftree representing a proof for " p is not true in C of P ". The proof is based on the idea that p cannot be true in C of P only if each rule whose head is p is either non-applicable or c-defeated which means that (1) its body can be proven not to be true or that (2) there is a competitor of which both the body and the head can be proven to be true. In particular, if the root of the overall sceptical prooftree has a positively (resp. negatively) adorned label p^+ then it represents a proof for " p is true in C of P " (resp. " p is not true in C of P ") in terms of a number of sceptical (sub)prooftrees.

Definition 11.

Let P be an ordered logic program and C a component of it. A sceptical prooftree in C of P is a finite tree where nodes are labeled by members of $\{p^+ \mid p \in B_P \cup \neg B_P\}$ and $\{p^- \mid p \in B_P \cup \neg B_P\}$. Moreover, each node n must satisfy one of the following conditions depending on its label:

- If n has label p^+ then there must be some rule r in $\text{ground}(C^*)$ with $H(r)=p$ such that n has a child labeled c^+ for each $c \in B(r)$. Moreover, for each competitor \hat{r} in $\text{comp}_{P|C}(r)$, n must have a child labeled d^- for some

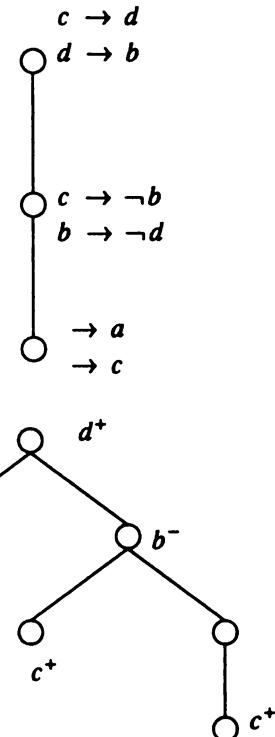
$$d \in B(\hat{r}) \cup H(\hat{r}).$$

- If n has label p^- then either
 - for every rule r in $\text{ground}(C^*)$ with $H(r)=p$, n must have a child labeled c^- for some $c \in B(r)$, or r must have a competitor \hat{r} in $\text{comp}_{P|C}(r)$ such that n has a child labeled d^+ for each $d \in B(\hat{r}) \cup H(\hat{r})$; or
 - n must have an ancestor n' with the same label (p^-) such that no node between n and n' has a + label.

A sceptical prooftree is said to be a sceptical prooftree of p^* in C of P if its root is labeled p^* . \square

Example 3.

The figure below illustrates an ordered logic program and a prooftree of d^+ in the bottom component C of this program.



\square

Definition 12.

Let P be an ordered logic program, C a component of it and p^* (where r is either + or -) an adorned ground literal. The existence of a sceptical prooftree of p^* in C of P is denoted by $P_C \vdash_c p^*$. Moreover, P_C^{c+} denotes $\{p \mid P_C \vdash_c p^+\}$, P_C^{c-} denotes $\{p \mid P_C \vdash_c p^-\}$ and $P_C^? = P_C^{c+} \cup P_C^{c-}$. P_C^{c+} is called the credulous extension of P in C . P_C^{c-} is called the credulous exclusion of P in C . \square

So, the credulous extension P_C^{c+} is the set of ground literals that, following the credulous approach, can be proven to be true in C of P and the credulous exclusion P_C^{c-} is the set of ground literals that can be proven not to be true in C of P while $P_C^{c?}$ contains those ground literals for which no proof exists.

As shown in Lae90c, P_C^{c+} and P_C^{c-} are well-defined: $P_C^{c+} \cap P_C^{c-} = \emptyset$ and P_C^{c+} is a consistent interpretation. ■

It turns out that the credulous extension P_C^{c+} is the sceptical c-partial model and that the credulous exclusion P_C^{c-} is the unfounded set w.r.t. the sceptical c-partial model.

Theorem 3.

Let P be an ordered logic program and C a component of it. Then $P_C^{c+} = S_P^c(\emptyset)$, and $P_C^{c-} = U_P^c|_C(S_P^c(\emptyset))$. ■

Note that the above theorem gives us a procedural top-down characterization of the sceptical c-partial model.

Another interesting result is that the sceptical proof theory is sound and complete w.r.t. the c-assumption-free c-partial model semantics.

Theorem 4. (Soundness)

Let P be an ordered logic program, C a component of it and M a c-assumption-free c-partial model for P in C . Then $P_C^{c+} \subseteq M$ and $M \cap P_C^{c-} = \emptyset$. ■

Theorem 5. (Soundness and Completeness)

Let P be an ordered logic program, C a component of it. The intersection of all c-assumption-free c-partial models is exactly the credulous extension. ■

A natural question to ask then is what is the relationship between the conservative analogues of credulous and sceptical c-partial models. This is the topic of the next section.

6. CONSERVATIVE VS. CREDULOUS APPROACH

6.1. PARTIAL VS. C-PARTIAL MODELS

In this section, we present the relationship between partial models and c-partial models.

Lemma 6.

Let P be an ordered logic program, C a component of it and I an interpretation for P in C . Then

- (a) the greatest c -unfounded set of P in C w.r.t. I is a subset of the greatest unfounded set of P in C w.r.t. I , i.e. $U_P^c|_C(I) \subseteq U_P|_C(I)$.
- (b) the greatest c -assumption set of P in C w.r.t. I is a subset of the greatest assumption set of P in C w.r.t. I , i.e. $A_P^c|_C(I) \subseteq A_P|_C(I)$. ■

Theorem 6.

Let P be an ordered logic program and C a component of it. Every partial model for P in C is a c-partial model for P in C . Moreover, every assumption-free partial model for P in C is a c-assumption-free c-partial model for P in C . ■

The next counterexample illustrates that the reverse is not necessarily true.

Example 4.

Reconsider P_5 . The empty set is a c-partial model for P_5 in C which is not a partial model for P_5 in C . □

6.2. STABLE VS. CREDULOUS MODELS

As for the relationship between stable partial models and credulous c-partial models, we have the result that each stable partial model can be extended to a credulous c-partial model.

Theorem 7.

Let P be an ordered logic program and C a component of it. Every stable partial model for P in C is a subset of some credulous c-partial model for P in C . ■

The reverse may be false as is illustrated next.

Example 5.

Reconsider P_5 . In Example 2, $\{p\}$ is the (only) stable partial model for P_5 in C , which can be extended to two credulous c-partial models for P_5 in C : $\{p, \neg q\}$ and $\{p, q\}$. The reverse does not hold, i.e. there can be credulous c-partial models that are not extensions of stable partial models. This is illustrated in P_5 by the credulous c-partial model $\{\neg p, \neg q\}$ model for P_5 in C , which is not an extension of any stable partial model for P_5 in C . □

6.3. WELL-FOUNDED VS. SCEPTICAL MODELS

As far as the relationship between well-founded and sceptical models is concerned, we find the following result.

Theorem 8.

Let P be an ordered logic program and C a component of it. The sceptical c-partial model for P in C is a subset of the well-founded partial model for P in C . ■

Example 6.

For P_5 , the sceptical c-partial model in C is the empty set, which is a subset of the well-founded partial model $\{p\}$. □

By definition, the sceptical c-partial model is a subset of the intersection of all credulous c-partial models. As is shown in the next example, this feature is not always true for the well-founded partial model.

Example 7.

For P_5 , the intersection of all credulous c-partial models in C is the empty set whereas the well-founded partial model in C is $\{p\}$. □

For a discussion on the computation of c-partial models, we refer to ^{Lae90c}, where it is shown how to effectively compute c-partial models, credulous c-partial models and the sceptical c-partial model.

7. CONCLUSION

The combination of the conservative and credulous approach to the semantics of ordered logic programs yields a hierarchy of partial models: at the bottom end we have the unique (extremely) sceptical model, which is included in the unique well-founded (partial) model. The well-founded partial model can be extended to yield (possibly) multiple stable partial models, which result from different (conservative) assumption-free choices (see ^{Lae90a}). The credulous approach allows us to extend these stable partial models further by making (conservative) assumptions.

It should be noted that the new concepts for ordered logic programs are proper generalizations of the corresponding concepts for classical logic programs by considering a special class of ordered logic programs, called ordered versions^{Lae90b} of logic programs: the well-founded^{Gel88a} resp. the stable partial models^{Gel88b, Sac90a} of a classical logic program coincide with the well-founded resp. the stable partial models of its ordered version. In addition, in the case

of ordered version of logic programs, the well-founded and the stable partial models coincide with the sceptical and the credulous partial models respectively.

References

- Ben00a. M. Ben-Jacob and M. Fitting, "Stratified and three-valued logic programming semantics," *Proc. 5th Int. Conf. and Symp. on Logic Programming*, pp. 1054-1068, 1988.
- Gel88a. A. Van Gelder, K. Ross, and J. S. Schlipf, "Unfounded Sets and Well-Founded Semantics for General Logic Programs," in *Proc. of the Symposium on Principles of Database Systems*, pp. 221-230, 1988.
- Gel88b. M. Gelfond and V. Lifschitz, "The Stable Model Semantics for Logic Programming," in *Proc. of the Intl. Conf. on Logic Programming*, pp. 1071-1079, 1988.
- Lae89a. E. Laenens and D. Vermeir, *A Fixpoint Semantics of Ordered Logic*, 1989. University of Antwerp, UIA Tech. Report 89-27
- Lae90a. E. Laenens and D. Vermeir, "A Fixpoint Semantics of Ordered Logic," *Journal of Logic and Computation*, vol. 1, no. 2, pp. 159-185, 1990.
- Lae90b. E. Laenens, D. Sacca, and D. Vermeir, "Extending logic programming," in *Proceedings of the SIGMOD conference*, pp. 184-193, 1990.
- Lae90c. E. Laenens, *Foundations of Ordered Logic*, 1990. PhD Thesis, University of Antwerp UIA
- Lae90d. E. Laenens and D. Vermeir, *Assumption-free semantics for ordered logic programs: on the relationship between well-founded and stable partial models*, 1990. University of Antwerp, Tech. Report 90-19
- Lae91a. E. Laenens and D. Vermeir, *On the Relationship between Well-Founded and Stable Partial Models*, 1991. Proc. of the Mathematical Fundamentals of Database and Knowledge Base Systems, to appear
- Llo87a. J.W. Lloyd, in *Foundations of Logic Programming*, Springer Verlag, 1987.
- Prz89a. T. Przymusinski, "Three-Valued Formalizations of Non-Monotonic Reasoning and Logic Pro-

gramming," *Proc. 1st Int. Conference on Principles of Knowledge Representation and Reasoning*, pp. 341-349, 1989.

Sac90a.

D. Sacca and C. Zaniolo, "Stable models and Non-determinism for logic programs with negation," *Proc. ACM Symp. on Principles of Database Systems*, 1990.

Ver89b.

D. Vermeir, D. Nute, and P. Geerts, "A logic for defeasible perspectives," in *Proc. of the 1988 Tübingen Workshop on Semantic Networks and Nonmonotonic Reasoning*, Vol. 1, pp. 1-27, SNS-Bericht 89-48, 1989.

Ver89a.

D. Vermeir, D. Nute, and P. Geerts, "A defeasible logic for multi-expert systems," in *Proc. of the Intl. Symposium on Computational Intelligence 89*, Elsevier Publ. Co., 1989.

You90a.

Jia-Huai You and Li Yan Yuan, "Three-Valued Formalization of Logic Programming: Is It Needed?," in *Proc. of the PODS'90 conference*, pp. 172-182, 1990.

Beyond Negation as Failure

Hector Geffner
 hector@ibm.com

T. J. Watson Research Center
 P. O. Box 704, Room H1-K10
 Yorktown Heights, N.Y. 10598

Abstract

After more than ten years of research in non-monotonic logics, the negation as failure rule (NAF) remains as the preferred device for capturing non-monotonic reasoning in AI programs. Non-monotonic logics have contributed to the analysis of NAF and other forms of reasoning but have not yet delivered devices as useful. Still, there are well known limitations in using negation as failure for representing default knowledge: negative information cannot be expressed and programs often loop. In this work we develop a new framework for logic programs which tackles these problems. The framework is structured in three parts: a model-theory, a sound and complete proof-theory, and various proof-procedures. The model-theory is based on a model-preference criterion which distinguishes atoms being true from atoms being *supported*. The proof-theory appeals to arguments (consistent sets of assumptions) which may conflict with each other and be *protected* from each other. Finally, the proof-procedures aim to provide efficient although incomplete approximations of the proof-theory. We then analyze how the resulting framework compares with accounts based on stable and well-founded models, and how it relates to non-monotonic TMSs.

1 Introduction

The first ten years of research in non-monotonic reasoning have produced a number of important results regarding the specification and content of defeasible knowledge bases. A number of formalisms are now available which allow us to speak with precision about revisable patterns of inference, and to encode and capture various forms of commonsense inference. However, in spite of these results, little has reached to

practitioners. People continue writing programs which perform non-monotonic inference and yet find most theorizing in non-monotonic reasoning irrelevant.

Nonetheless, the existing devices for capturing non-monotonic reasoning in programs are far from perfect. For example, if one uses PROLOG's negation as failure, one cannot express that two predicates are incompatible, and must be especially careful to avoid loops (e.g., (Kowalski and Sergot, 1986)). Similar problems occur in truth-maintenance systems. This raises the question of whether new architectures can be developed in which the computational concerns that make devices like NAF and TMSs useful in practice, and the expressive power that makes non-monotonic formalisms adequate in theory, are combined.

This work is an exploration on that theme. We develop a new form of negation as failure which, while suitable for a larger class of programs than Clark's NAF, retains two of its most salient features: an intuitive behavior, and a definite-clause model of computation. The new negation as failure rule, can also handle explicit negative information and avoids the problems caused by the presence of loops.

As we illustrate here and argue in more detail elsewhere (Geffner, 1991), the addition of a more powerful form of negation to logic programs, together with the ability to handle negative information results into an appealing representational framework for default reasoning; a framework which does well in capturing the intuitions of the user, both in terms of what conclusions are sanctioned and how they are obtained.

This paper is organized as follows. In section 2 we discuss some problems with negation as failure and introduce a slight variation of NAF, called X^1 NAF, which addresses them. Basically, X^1 NAF removes the procedural aspects of Clark's negation by introducing the logical notion of *support*. The new form of negation as failure coincides with the well-founded semantics, and thus is sound and complete for the class of stratified programs, and remains sound within the larger class of programs with stable models. Yet, X^1 NAF

possesses serious limitations. One of them is that it cannot handle negative information. In section 3 we introduce another extension of negation as failure, called X²NAF, which overcomes these problems. We then discuss the assumptions underlying X²NAF, and what X²NAF computes and *fails* to compute. We do so by developing a novel semantics for logic programs and a sound and complete proof-theory. We show that X²NAF provides a sound approximation of the proof-theory, provided suitable conditions hold. This approximation is then improved in section 5. Related work is discussed in section 6 and throughout the paper. Section 7 summarizes the main contributions.

2 Negation as Failure

The negation as failure rule originates in the devices used in PLANNER (Hewitt, 1972) and PROLOG (Rousell, 1975) for making assumptions about negative information. Clark (1978) was the first to give both a precise description of the rule and a logical account of its behavior. Basically, Clark's negation as failure (NAF) extends a sound and complete strategy for programs with no negation, into an strategy for programs with negation. The additional step permits us to derive *negative* literals of the form $\neg\beta$ when the derivation of the *positive* literal β (finitely) fails.¹ As Clark showed, under mild assumptions, the literals computed by the negation as failure rule are sound with respect to the logical theory that results from 'completing' the program.

Nonetheless, the interpretation of negation that results from Clark's account has some peculiarities. Loops, for example, often render a behavior which is hard to justify on logical grounds. For example, the addition of what should be a harmless rule, $p \leftarrow p$, sometimes changes the 'meaning' of a program. Thus, a program $P_0 = \{q \leftarrow \neg p\}$ sanctions q , but a program $P_1 = \{q \leftarrow \neg p; p \leftarrow p\}$ does not.

More recent accounts have cured these anomalies. For the family of *stratified* programs, for example, it has been found that there is a unique Herbrand model, which faithfully and completely captures the intention of the programmer (Apt *et al.*, 1987). Such a distinguished model, indeed, is sanctioned by a number of accounts each resting on slightly different intuitions (see (Przymusinska and Przymusinski, 1989) for a review).

However, while these accounts provide a precise interpretation of logic programs with negation, their applicability to problems involving defaults remains limited. Consider for example the program that results from expressing two conflicting defaults, say "quakers

¹For an introduction to logic programming, see (Lloyd, 1984).

are pacifists" and "republican are not pacifists":

$$\begin{aligned} \text{pacifist}(z) &\leftarrow \text{quaker}(z) \wedge \neg \text{ab}_1(z) \\ \text{non_pacifist}(z) &\leftarrow \text{republican}(z) \wedge \neg \text{ab}_2(z) \\ \text{ab}_2(z) &\leftarrow \text{quaker}(z) \wedge \neg \text{ab}_1(z) \\ \text{ab}_1(z) &\leftarrow \text{republican}(z) \wedge \neg \text{ab}_2(z) \end{aligned}$$

The program contains a negative loop relating the two 'abnormalities' ab_1 and ab_2 , and thus, is not stratified. This rules out procedures like SLS-resolution (Przymusinski, 1987), and causes the most commonly used SLDNF (Lloyd, 1984) to loop. (e.g., when queried about whether a republican quaker is a pacifist). Semantics based on stable or well-founded models (Gelfond and Lifschitz, 1988, Van Gelder *et al.*, 1988), on the other hand, correctly define the behavior of these programs, but fail to provide effective proof-procedures for delivering the answers.²

In this paper we aim to make the framework of logic programs with negation into a more adequate framework for default reasoning. For that we will develop a semantics and proof-procedures which will allow us to deal with the example above in the following simpler form:

$$\begin{aligned} \text{pacifist}(z) &\leftarrow \text{quaker}(z), \neg \text{ab}_1(z) \\ \text{non_pacifist}(z) &\leftarrow \text{republican}(z), \neg \text{ab}_2(z) \\ &\leftarrow \text{pacifist}(z), \text{non_pacifist}(z) \end{aligned}$$

In order to motivate the new ideas, we will start with Clark's negation as failure, which we will later recast into a more convenient form.

2.1 From NAF to X¹NAF

Clark's negation as failure can be usefully understood in terms of the following *pseudo*-definition:³

- D1. a *negative* literal $\neg p$ is derivable, if the positive literal p is *not* derivable
- D2. a *positive* literal p is derivable, if the program contains a rule $p \leftarrow L_1, L_2, \dots, L_n$, and each literal L_i is derivable

We say that this is a *pseudo*-definition because it does not define the 'derivability' relation at all. D1 and D2 depend on each other, and D1 in particular appeals to the *negation* of the notion we are trying to define. Clark's negation as failure thus *interprets* this

²This remark is not completely fair. Although the procedures developed for the well-founded semantics (e.g., (Ross, 1989)) will loop in this program, they can in principle be extended with (albeit incomplete) loop trapping mechanisms. The same extension, however, will not work for SLDNF or SLS-resolution.

³For simplicity, we assume that the program under consideration is ground (i.e., does not contain any variables).

pseudo-definition, with the added provision that positive literals whose derivations (finite) fail are taken to be non-derivable. The same is true about PROLOG interpreters. The problem of this approach though, is that introduces unnecessary loops. In a program $P = \{r \leftarrow \neg p ; p \leftarrow q ; q \leftarrow p\}$, for example, we expect r to be derivable, but Clark's NAF loops.

It is possible, nonetheless, to convert the pseudo-definition D1-D2 into a legitimate, inductive definition. The key notion for that translation will be the notion of *support* which we define below. Roughly, a set of negative literals $\neg \beta_i$ will support a positive literal p , when p can be derived from D1 and D2 *assuming* that none of the atoms β_i is derivable. We formalize this notion after the following preliminary definitions.

The programs we consider are collection of rules of the form $A \leftarrow L_1, \dots, L_n$ where A is an atom, L_1, \dots, L_n are literals (i.e., atoms or negated atoms) and $n \geq 0$. Rules of this form are notational variants of the formula $L_1, \dots, L_n \Rightarrow A$, so that any program can be viewed as a set of first-order formulas. Variables in such rules are implicitly universally quantified. P^* will be used to denote the ground program obtained by substituting each rule in P by all its ground instances.

Now, for a set Δ of ground negative literals, let P_Δ stand for the collection of ground rules in P enabled by Δ , i.e. the rules in P^* all whose negative literals belong to Δ . Furthermore, let P_Δ^+ stand for the positive ground program obtained from P_Δ by removing all negative literals.⁴ Then, the support relation is defined as follows:

Definition 1 (Support) *A collection of ground negative literals Δ supports an atom p in a program P , if p is a logical consequence of the definite program P_Δ^+ .*

For example if P is given by the rules:

$$\begin{aligned} c(x) &\leftarrow f(x), \neg a(x), \neg b(x) \\ d(x) &\leftarrow \neg a(x) \\ f(x) &\leftarrow d(x) \end{aligned}$$

and $\Delta = \{\neg a(t)\}$, then P_Δ^+ is given by the program $\{d(t) \leftarrow ; f(t) \leftarrow d(t)\}$. As a result, the atoms $d(t)$ and $f(t)$, which are logical consequences of P_Δ^+ , are supported by Δ .

Note that if an atom p is not supported by a set of negative literals, p will not be derivable by negation as failure. Furthermore, p will still *not* be derivable by negation as failure, if the only supports p has, all involve literals $\neg q$ whose complements q are derivable. We can thus recast the pseudo-definition D1-D2 in the following form:

⁴This transformation is similar to the one used in (Gelfond and Lifschitz, 1988).

D1'. a negative literal $\neg p$ is derivable, if every support Δ' for p contains a literal $\neg q$ such that q is derivable

D2'. a positive literal p is derivable, if it is supported by a set of derivable literals $\neg p$;

Such form can be further unfolded yielding a legitimate, *inductive*, definition of negation as failure:

D1''. a negative literal $\neg p$ is derivable, if every support Δ' for p contains a literal $\neg q$ such that q is supported by a set of derivable literals $\neg p$;

D2''. a positive literal p is derivable, if it is supported by a set of derivable literals $\neg p$;

Note that the derivation of positive literals plays now no role on the derivation of negative literals. The latter, on the other hand, completely determine the former.

We will be referring to negative ground literals as *assumptions*. Line D1'' thus provides us with the conditions under which assumptions can be adopted. Line D2'', on the other hand, maps those assumptions into atoms via the support relation. For example, in the program $P = \{r \leftarrow \neg p ; p \leftarrow q ; q \leftarrow p\}$, the assumptions $\neg p$ and $\neg q$ are derivable, as neither p or q is supported. Furthermore, since $\neg p$ supports r , then the atom r is derivable as well.

The first proof-procedure we will study, called X¹NAF, is a slight variation of the definition D1''-D2'' above. X¹NAF accepts a *list of assumptions* and tell us whether each element in the list is derivable. The idea of the list is to make explicit the dependences among assumptions. For example, if the assumption α_1 can be derived by means of D1'' and D2'' only after an assumption α_2 is derived, lists containing α_1 will be derivable only if α_2 precedes α_1 in the list. In the program $P = \{c \leftarrow \neg b ; b \leftarrow \neg a\}$, for instance, this means that the lists $\{\neg a\}$ and $\{\neg a, \neg c\}$ will be derivable, while the list $\{\neg c\}$ will not. We will say more below about how such lists are constructed to derive propositions of interest.

We use the notation $\Delta_i = \Delta_{i-1} + \alpha_i$ to denote the list obtained by appending the element α_i to the list Δ_{i-1} . When the order in a list is not relevant, we treat lists as sets. X¹NAF is then defined as follows:

Definition 1 (X¹NAF)

1. $\Delta_0 = \emptyset$ is derivable from P
2. $\Delta_i = \Delta_{i-1} + \alpha_i$ is derivable from P if
 - 2.1 Δ_{i-1} is derivable from P , and
 - 2.2 every set Δ' which supports α_i , contains an assumption $\neg \beta_j$, such that Δ_{i-1} supports β_j

The definition is extended to literals in a straightforward way: the assumptions derivable by X¹NAF are

the assumptions which belong to some derivable list, while the ground atoms derivable by X¹NAF are the atoms *supported* by collections of derivable assumptions $\neg\beta_i$.

Like Clark's NAF, X¹NAF involves computing with definite programs only. X¹NAF however, leaves many control decisions unspecified: how to construct the lists Δ_i , which counterarguments Δ' to try first, and so on. It is fair to say, indeed, that X¹NAF is closer in form to the so called *argument systems* developed for defeasible reasoning (e.g., (Pollock, 1987, Loui, 1987)) than to the query evaluation procedures for logic programs. Line 2.2, for example, can be understood as saying that in order to accept Δ_i , every 'argument' Δ' that *rebuts* Δ_i (i.e., Δ' supports β_i) must be *defeated* by Δ_{i-1} (i.e., $\Delta' + \Delta_{i-1}$ must be self-defeating).⁵ This view actually provides the right intuition for adding to X¹NAF a layer of control: if we want to prove p , then we will try to find a list Δ which supports p . If no Δ' rebuts Δ , then we are done; otherwise we need to *prepend* to Δ a set Δ'' of assumptions which defeats Δ' , and repeat the process for $\Delta'' + \Delta$.

The following theorems relate X¹NAF to other proposals in the literature.⁶

Theorem 1 *If P is stratified, then an atom p is derivable by X¹NAF if and only if $p \in M_P$, where M_P is the canonical Herbrand model of P.*⁷

Theorem 2 *An atom p is derivable by X¹NAF if and only if p is true in the well-founded (possibly partial) model of P.*

Theorem 3 *If p is derivable by X¹NAF then p belongs to all the stable models of P.*

Note that theorems 2 and 3 do not require programs to be stratified. Theorem 2 says that X¹NAF and the well-founded models semantics coincide in an important sense. This is most apparent when one compares the definition of X¹NAF with the formulation of well-founded models based on alternating fixed-points (Van Gelder, 1989).

3 Handling Negative Information

In spite of these properties, however, X¹NAF is not the proof-procedure we are looking for. We discuss

⁵Note that it is sufficient to consider the *minimal* such Δ'' 's only.

⁶The notion of stratification can be found in (Apt et al., 1987). The semantics based on well-founded models and stable models were introduced in (Van Gelder et al., 1988) and (Gelfond and Lifschitz, 1988, Fine, 1989) respectively.

⁷Proofs of the theorems will appear in the full version of this paper to be available from the author.

three problems with X¹NAF, before introducing a more powerful procedure which addresses them.

Inconsistent Rebuttals. One of the problems of X¹NAF pops up in the program P_1 :

$$\begin{aligned} c &\leftarrow \neg a, \neg b \\ b &\leftarrow \neg a \\ a &\leftarrow \neg b \end{aligned}$$

P_1 can be understood as asserting that c follows from the assumptions $\neg a$ and $\neg b$, but that one of these assumptions must be false. It is natural to expect then that $\neg c$ will be derivable; however, if we try to prove the list $\Delta_1 = \{\neg c\}$ by means of X¹NAF we would find that c is supported by the set $\Delta = \{\neg a, \neg b\}$ and thus $\{\neg c\}$ is not sanctioned. Intuitively, this is wrong as we are rebutting the list Δ_1 with an assumption set Δ which is *inconsistent*. Δ is inconsistent because it contains the assumptions $\neg a$ and $\neg b$, but supports their complements a and b . We say in that case that Δ is *self-defeating*. More generally,

Definition 2 *A set of assumptions Δ is self-defeating if it supports an atom β such that $\neg\beta \in \Delta$.*

The problem of inconsistent rebuttals can thus be dealt with by requiring that rebuttals be non-self-defeating. Note that the computation of self-defeat, like the computation of support, still involves definite clauses only.

Indefinite Defeat. A similar problem to the one above arises in the program P_2 :

$$\begin{aligned} c &\leftarrow \neg a, \neg b \\ b &\leftarrow \neg d, \neg a \\ a &\leftarrow \neg d, \neg b \end{aligned}$$

Now X¹NAF derives the list $\Delta_1 = \{\neg d\}$ but does not derive the list $\Delta_2 = \Delta_1 + \{\neg c\}$. This is due to the set $\Delta = \{\neg a, \neg b\}$ which supports c . Although Δ is no longer inconsistent in this program, Δ is inconsistent with the assumptions derivable from the program, in this case $\Delta_1 = \{\neg d\}$. However, although Δ_1 makes the rebuttal Δ inconsistent, it fails to support either a or b . Note, however, that if $\neg a$ is accepted, Δ_1 would support b , and if $\neg b$ is accepted, Δ_1 would support a . In either case or if neither $\neg a$ nor $\neg b$ is accepted, thus $\neg c$ should be derivable.

The key to handle programs like P_1 lies again on the notion of self-defeat. While in line 2.2 of the definition of X¹NAF, we required Δ_{i-1} to support the complement of one of the assumptions in Δ' in order to defeat Δ' , we will now require only that the union $\Delta' + \Delta_{i-1}$ be *self-defeating*. Since Δ_{i-1} has been derived already, this would imply that the complement of one of the assumptions in Δ' will be supported, once we are willing to accept a consistent subset of Δ' . In P_2 , for example, $\Delta_1 = \{\neg d\}$ will defeat the rebuttal $\Delta = \{\neg a, \neg b\}$

because $\Delta_1 + \Delta$ is self-defeating (it supports both a and b). As a result, the list $\Delta_2 = \Delta_1 + \{\neg c\}$ will be derivable.

Negative Information. A more difficult problem than the two above is the handling of negative information. The need for expressing negative information arises in many AI tasks. When reasoning about change, for instance, say in scenarios like the Yale Shooting Problem, we want to be able to say things like that the gun is not loaded, that dead and alive are incompatible and so on. Still the language of definite clauses with negation as failure does not allow us to do so.

The next step then is to extend the expressive power of logic programs to accommodate negative information. We will do so by means of *negative constraints* of the form

$$\leftarrow A_1, A_2, \dots, A_n$$

where each literal A_i is an atom. The meaning of such constraints corresponds exactly to their logical meaning: one of the atoms A_i must be *false*. Thus we will write $\leftarrow \text{loaded}(5)$ to express that the gun was not loaded at time $t = 5$, $\leftarrow \text{dead}(t), \text{alive}(t)$ to express that ‘dead’ and ‘alive’ are incompatible, and so on.

The incorporation of constraints in the language requires us to define what the meaning of program with constraints is. Familiar semantics like stable models don’t apply because most programs with constraints lack stable models. Still we do have intuitions about what the right behavior of such programs ought to be. In the next section we will formalize such intuitions. We will then show that the proof-procedure below complies with the formalization.

The proof-procedure $X^2\text{NAF}$ operates on a program P augmented with a collection of negative constraints N . The main difference between $X^2\text{NAF}$ and $X^1\text{NAF}$ besides those mentioned above, is that constraints provide an additional source of rebuttals. For example, if an assumption α supports an atom p and N contains a constraint $\leftarrow p$, it is reasonable *not* to derive α . More generally, we will have an assumption set Δ rebut an assumption α , when Δ extended with α supports each of the atoms p_i of a ground instance $\leftarrow p_1, \dots, p_n$ of a constraint in N . We will abbreviate that by saying that $\Delta + \{\alpha\}$ supports \overline{N} .

The introduction of constraints does not affect the definitions of support and self-defeat which are still determined by P alone. Yet there are two sources of inconsistency now: a set Δ of assumptions may be inconsistent because it is self-defeating or because it supports \overline{N} . In either case we will say now that Δ is *inadmissible*; and otherwise, that Δ is *admissible*. Note that admissibility, like support and self-defeat, is still computed in terms of definite programs only.

Provided with these notions and a program $P+N$, $X^2\text{NAF}$ is inductively defined as follows:

Definition 2 ($X^2\text{NAF}$)

1. $\Delta_0 = \emptyset$ is derivable from $P+N$
2. $\Delta_i = \Delta_{i-1} + \neg\beta_i$ is derivable from N if
 - 2.1 Δ_{i-1} is derivable from $P+N$, and \forall admissible Δ'
 - 2.2.a if Δ supports β_i , $\Delta' + \Delta_{i-1}$ is self-defeating, and
 - 2.2.b if $\Delta' + \{\neg\beta_i\}$ supports \overline{N} , $\Delta' + \Delta_i$ is self-defeating

The definition is extended to literals in the straightforward way. A negative literal α is derivable from $P+N$ by $X^2\text{NAF}$, if α occurs in some list derivable by $X^2\text{NAF}$, while a positive literal p is derivable from $P+N$ by $X^2\text{NAF}$, if there is a collection Δ of negative literals that supports p such that each α_i in Δ is derivable.

$X^2\text{NAF}$, like $X^1\text{NAF}$, can also be viewed as an argument system. Indeed, $X^2\text{NAF}$ involves two types of rebuttals. Line 2.2.a, for example, can be understood as stating that if Δ' *strongly rebuts* Δ_i (i.e., Δ' supports β_i) then the assumptions preceding $\neg\beta_i$, i.e. Δ_{i-1} , must *defeat* Δ' . Line 2.2.b, on the other hand, says that if Δ' *weakly rebuts* Δ_i (i.e., $\Delta_i + \{\neg\beta_i\}$ supports \overline{N}), then we are allowed to defeat Δ' with the entire list Δ_i , with the assumption $\neg\beta_i$ included.

The distinction between the two forms of rebuttals can be illustrated by means of the program P_2+N :

$$\begin{aligned} b &\leftarrow \neg a \\ c &\leftarrow \neg b, \neg a \\ &\leftarrow c \end{aligned}$$

The assumption set $\Delta = \{\neg b\}$ rebuts the list $\Delta_1 = \{\neg a\}$ because $\Delta + \Delta_1$ supports c , and hence \overline{N} . The set Δ , however, does not support the atom a . As a result, the definition of $X^2\text{NAF}$ permits us to establish Δ_1 by showing $\Delta + \Delta_1$ to be self-defeating. This follows from the rule $b \leftarrow \neg a$. If the program P_2 is extended with another rule $a \leftarrow \neg b$, however, the assumption $\neg a$ would no longer be derivable, as $\Delta = \{\neg b\}$ would support a , and hence, the definition of $X^2\text{NAF}$ would require that Δ be defeated by the assumptions preceding $\neg a$.

The characterization of what $X^2\text{NAF}$ computes will be given in the next section. Before proceeding with such a characterization, we consider an example in detail.

Example 1 The program $P+N$ below represents a variation of the Yale shooting scenario (Hanks and McDermott, 1987):

$$\begin{aligned} \text{alive}(s(t)) &\leftarrow \text{alive}(t), \neg \text{ab}_1(t) \\ \text{loaded}(s(t)) &\leftarrow \text{loaded}(t), \neg \text{ab}_2(t) \\ \text{dead}(s(t)) &\leftarrow \text{shoot}(t), \text{loaded}(t), \neg \text{ab}_3(t) \\ \text{ab}_1(t) &\leftarrow \text{shoot}(t), \text{loaded}(t), \neg \text{ab}_3(t) \end{aligned}$$

```

loaded(0) ←
alive(1) ←
shoot(1) ←
← alive( $t$ ), dead( $t$ )

```

The predicates ab_i 's stand for 'abnormalities,' s stands for the successor function, and positive integers n are used as abbreviations for n applications of the successor function to 0. It is possible to check then that the list $\Delta_2 = \{\neg\text{ab}_2(0), \neg\text{ab}_3(1)\}$ is derivable from $P+N$, and as a result, so is the atom $\text{dead}(2)$ which it supports. In this regard, $X^2\text{NAF}$ behaves no different than Clark's negation as failure. Eshghi and Kowalski (1989) showed indeed that the Yale shooting problem does not arise if the scenario is represented in a logic program with negation as failure. However, it can be argued that those solutions 'cheat' in the following sense: they get the right behavior because of the built-in inability of negation as failure to consider the contrapositives of the rules in the program. This is not the case with $X^2\text{NAF}$. For example, if the person is found alive after the shooting, say $\text{alive}(2)$, the conclusion $\text{dead}(2)$ would be retracted, as the list Δ_2 would be rebutted by the empty assumption set. Moreover, in that scenario, we would not conclude that the gun remained loaded ($\text{loaded}(1)$) as the assumption involved, $\neg\text{ab}_2(0)$, is now rebutted by the set $\Delta' = \{\neg\text{ab}_3(2)\}$. Thus $X^2\text{NAF}$, unlike NAF, can reason both backward and forward.

4 Semantics

Preferential Support. In this section we will focus on the semantics of the proof-procedures discussed so far, trying to uncover the assumptions they embed, and what they compute and *fail* to compute. The semantics is articulated in two parts: first, the program P and the constraints N determine what the (Herbrand) *models* of the program $P+N$ are, then those models are ranked according to a preference criterion which distinguishes atoms being *true* from atoms being *supported*. We refer to the resulting semantics as *preferential support*.⁸

We start by extending the notion of support to models. We say that an atom p is *supported in a model M* if there is a set Δ of assumptions which supports p such that each $\neg\beta_i$ in Δ holds in M (i.e., $\beta_i \notin M$). Note that an atom may be true in a model and yet be unsupported, although all supported atoms must be true. For example, if $P = \{p \leftarrow \neg q\}$ and N is empty, then $M = \{q\}$ is a model of $P+N$ in which q is true but not supported. On the other hand, in the

⁸The semantics and proof-theory below correspond to special cases of the the semantics and proof-theory of the 'causal theories' discussed in my (1990). More on this relation, in section 6.

model $M' = \{p\}$, p is both true and supported (by $\Delta = \{\neg q\}$).

The preference relation on models is defined as follows. Let $X[M]$ denote the collection of atoms supported by M . Then we say that a model M is *as preferred as* a model M' , written $M \preceq M'$, if $M - M' \subseteq X[M]$, i.e. if every atom in M but not in M' is supported by M . This preference relation is neither antisymmetric nor transitive, and hence certain care is required in defining what the preferred models of a program are. We thus introduce the notion of a *basis* B , as a collection of models which obey the following two conditions. First, for each model M' not in B there must be a model M in B such that $M \preceq M'$. Second, if M is a model in B and M' is a model not in B such that $M' \preceq M$, then B must contain a model M'' such that $M'' \subset M'$. Intuitively, the first condition says a basis must be *exhaustive*, the second, that it must be *closed*. It can be shown that if B' and B'' are two bases, then the intersection $B' \cap B''$ is a basis as well. Thus, any program $P+N$ possesses a *minimal* basis B^* ; we call the models in B^* the *preferred models* of $P+N$. It can be shown that preferred models are always *minimal*.

For the program $P = \{q \leftarrow \neg p\}$ above, for example, $M' = \{q\}$ is the only preferred model. In general, if M is a stable model, M and $X[M]$ will coincide and thus M will always be a preferred model. However, a program may have preferred models which are not stable, even when the program has stable models. Two examples of such programs are:⁹

$P_1 :$ <table style="margin-left: 20px;"> <tr><td>$a \leftarrow \neg b$</td></tr> <tr><td>$b \leftarrow \neg a$</td></tr> <tr><td>$p \leftarrow \neg a$</td></tr> <tr><td>$p \leftarrow \neg p$</td></tr> </table>	$a \leftarrow \neg b$	$b \leftarrow \neg a$	$p \leftarrow \neg a$	$p \leftarrow \neg p$	$P_2 :$ <table style="margin-left: 20px;"> <tr><td>$a \leftarrow \neg b$</td></tr> <tr><td>$b \leftarrow \neg a$</td></tr> <tr><td>$p \leftarrow \neg a$</td></tr> <tr><td>$b \leftarrow \neg p$</td></tr> </table>	$a \leftarrow \neg b$	$b \leftarrow \neg a$	$p \leftarrow \neg a$	$b \leftarrow \neg p$
$a \leftarrow \neg b$									
$b \leftarrow \neg a$									
$p \leftarrow \neg a$									
$p \leftarrow \neg p$									
$a \leftarrow \neg b$									
$b \leftarrow \neg a$									
$p \leftarrow \neg a$									
$b \leftarrow \neg p$									

In both cases, $M = \{b, p\}$ is the single stable model. However, $M' = \{a, p\}$ is also a preferred model for both P_1 and P_2 , and $M'' = \{a, b\}$ is a preferred model of P_2 as well.

Proof-Theory. The semantics of preferential support also possesses a sound and complete proof-theory (different than the proof-procedures above which exploit simplifying assumptions). Indeed, in contrast to the arguments one often hears about model-theory being the 'true and only carrier' of meaning, I find the proof-theory of preferential support more intuitive and compelling than its model-theory.

Let us say that a set of assumptions Δ constitutes an *argument* in a program $P+N$ when Δ is logically consistent with $P+N$. Furthermore, let two arguments Δ and Δ' be in *conflict* when their union $\Delta + \Delta'$ is inconsistent with $P+N$. We say then that an argument Δ is *protected* from a conflicting argument Δ' if there is a collection of assumptions Δ_A in $\Delta' - \Delta$ such that 1) the

⁹ P_1 is from Van Gelder et al. (1988).

collection of assumptions Δ_B , $\Delta_B = \Delta + \Delta' - \Delta_A$, is logically consistent with $P+N$, and 2) Δ_B supports the complement β_i of each assumptions $\neg\beta_i$ in Δ_A . Intuitively, Δ is protected from a conflicting Δ' when Δ has a consistent superset Δ_B in which the complement of each assumption in Δ' but not in Δ_B , is 'explained' (i.e., supported).

Now, let us say that an argument Δ is *stronger* than a conflicting argument Δ' when Δ is protected from Δ' and Δ' is not protected from Δ , and let Δ be *stable* when it is stronger than every conflicting argument Δ' . Then, the following result can be shown:

Theorem 4 *If Δ is a stable argument in a program $P+N$, then Δ holds in each of the preferred models of $P+N$.*

For example, if P_3+N is given by the clauses:

```
c ← ¬a
d ← ¬b
f ← ¬b, ¬c
← f
```

it can be shown that the argument $\Delta = \{\neg a, \neg b\}$ is stable. First, note that neither a or b can be supported and thus no conflicting argument will be protected from Δ . Moreover, any argument Δ' in conflict with Δ must contain one of the assumptions $\neg c$ or $\neg d$. Thus, let Δ_A stand for one of the sets $\{\neg c\}$, $\{\neg d\}$ or $\{\neg c, \neg d\}$, according to whether Δ' contains the assumption $\neg c$, $\neg d$, or both. Then it is simple to show that the set $\Delta_B = \Delta - \Delta_A$ is consistent and supports each of the atoms whose negations appear in Δ_A . In other words, Δ is stable. Theorem 4 then says that Δ must hold in every preferred model of $P+N$. This can now be verified as $M = \{c, d\}$ is the only preferred model of $P+N$.

Stable arguments provide us with sufficient conditions under which a proposition is sanctioned by preferential support. In order to obtain a complete proof-theory we need an additional refinement. Let us say that a collection C of arguments $\Delta_1, \dots, \Delta_n$ is a *stable cover*, when for any argument Δ' which is in conflict with every argument Δ_i in C , two conditions hold: first, Δ' is not protected from any Δ_i in C , and second, some Δ_i in C is protected from Δ' . Then the following syntactic characterization of preferential support results:

Theorem 5 *A proposition p is sanctioned by preferential support in a program $P+N$ if and only if there is a stable cover $C = \{\Delta_1, \dots, \Delta_n\}$ such that p is a logical consequence of $P+N+\Delta_i$, for $i = 1, \dots, n$.*

In a logic programming framework, however, we are normally not interested in all consequences of a program but only on those which are *supported*. The result above can thus be cast as follows:

Theorem 6 *An atom p is supported by every preferred model of $P+N$ if and only if there is a stable cover $C = \{\Delta_1, \dots, \Delta_n\}$, such that each Δ_i , $i = 1, \dots, n$, supports p in $P+N$.*

Note that if an argument Δ is stable, then the cover $C = \{\Delta\}$ is stable as well. However, stable covers capture patterns of inference which rely on *disjunctive* collection of assumptions which stable arguments cannot capture. For example, in the program $P = \{c \leftarrow b ; c \leftarrow a ; b \leftarrow \neg a ; a \leftarrow \neg b\}$, the cover $C = \{\Delta_1, \Delta_2\}$ with $\Delta_1 = \{\neg a\}$ and $\Delta_2 = \{\neg b\}$ is stable and supports the atom c . On the other hand, neither Δ_1 nor Δ_2 is stable.

X²NAF. Provided with the model and proof-theory of preferential support we can now provide a characterization of what X²NAF computes. X²NAF computes a special class of stable arguments that we call *linear*. A linear stable argument is a stable argument Δ which can be ordered as a list $\Delta_n = \{\alpha_1, \dots, \alpha_n\}$ such that each prefix $\Delta_i = \{\alpha_1, \dots, \alpha_i\}$, for $i = 1, \dots, n$, is stable. For example, the argument Δ in the program P_3+N above, is linearly stable, as Δ can be written as a list $\Delta_2 = \{\neg a, \neg b\}$, and both $\Delta_1 = \{\neg a\}$ and Δ_2 are stable.

However, not every stable argument is linear. Consider for instance the program P_4+N' obtained from P_3+N by adding the following two clauses:

```
g ← ¬a, ¬d
      ← g
```

By arguments similar to the ones above it is possible to show that Δ remains stable; Δ , however, is no longer *linearly* stable. Indeed, neither $\Delta_1 = \{\neg a\}$ nor $\Delta'_1 = \{\neg b\}$ is stable: Δ_1 , because is not protected from the conflicting argument $\Delta_3 = \{\neg d\}$, while Δ'_1 , because is not protected from the conflicting argument $\Delta_4 = \{\neg c\}$. Being Δ linearly stable in P_3+N but not in P_4+N' , X²NAF computes Δ in the former program but not in the second.

There are programs, however, in which X²NAF computes non-linear arguments and even non-arguments as well (i.e., inconsistent sets of assumptions). For instance in the program $P_5 = \{b \leftarrow \neg a, \neg b\}$ X²NAF sanctions the list $\Delta_1 = \{\neg a, \neg b\}$ which is inconsistent with P_5 . Similarly, in the program $P_6+N = \{c \leftarrow a ; b \leftarrow \neg a ; b \leftarrow \neg c\}$, X²NAF sanctions the list $\Delta_2 = \{\neg c\}$ which is also inconsistent. Fortunately, however, it is possible to characterize a class of programs in which the behavior of X²NAF is sound. These are programs which comply with the two properties below. We define the properties first, and then discuss their significance.

First, let us say that a predicate p occurs *negated* in a program $P+N$, if $P+N$ contains a rule with a negative

literal $\neg p(x)$ in its body, where x denotes a tuple of terms. Let us also say that a predicate p occurs *unnegated* in a program $P+N$ if $P+N$ contains a rule or constraint with a *positive* literal $p(x)$ in its body. Then we say that a program is *open* if no predicate occurs both negated and unnegated. The program P_5 above, for example, is open, while the program P_6+N is not (a occurs both negated and unnegated).

Now, let P_Δ stand for the collection of ground rules in P *enabled* by Δ , i.e. the rules in P^* all whose negative literals belong to Δ (see section 2). Then we will say that a program P is *completely stable* if for any Δ , the subprogram P_Δ has a stable model.

Finally, let *default programs* $P+N$ be programs which are *open* and in which P is *completely stable*. Then, we can prove the following:

Theorem 7 *If $P+N$ is a default program, then a list of assumptions $\Delta_n = \{\alpha_1, \dots, \alpha_n\}$ is derivable by X^2NAF if and only if each subset $\Delta_i = \{\alpha_1, \dots, \alpha_i\}$, $i = 1, \dots, n$, is stable.*

In other words, in the context of default programs, X^2NAF computes all and only linear stable arguments.

Let us now elaborate on the meaning of the constraints imposed on the class of default programs. Regarding the openness condition, most ‘abnormality’ default theories are open: abnormalities occur negated in the body of rules, and no other predicates are assumed false by default. Negative literals, like $\neg pacifist(x)$, should thus be rewritten as *positive* literals $\text{non_pacifistic}(x)$, and asserted to be incompatible with the literals $\text{pacifist}(x)$. More generally, a non-open program can be ‘opened’ by introducing auxiliary predicates ξ and rewriting rules $\gamma \leftarrow \alpha_1, \dots, \alpha_n, \neg \beta_1, \dots, \neg \beta_m$ as $\gamma \leftarrow \alpha_1, \dots, \alpha_n, \neg \xi$ together with $\xi \leftarrow \beta_i$, for $i = 1, \dots, m$. The open and the un-opened version of a program will certainly have a different semantics, although it is possible that programmers really mean the ‘opened’ versions of the programs they write. Few logic programmers, I think, will be surprised by the conclusion $\neg c$ in the program P_5+N above; indeed, while $\neg c$ is inconsistent in P_5+N , it is a legitimate conclusion of its ‘open’ version.

The condition of complete stability is more critical, although perhaps not less reasonable. First of all, all stratified programs are completely stable. Non-completely stable programs, on the other hand, usually contain spurious rules like $b \leftarrow \neg a, \neg b$ which cannot provide support. They are thus *pure constraints*, and as such, we’d like to argue, should be placed in N rather than in P .¹⁰

¹⁰Since X^2NAF expects only positive atoms in N , a constraint like $\leftarrow \neg a, \neg b$ should be rewritten as a rule $c \leftarrow \neg a, \neg b$ together with the constraint $\leftarrow c$. That rewriting is not required by the semantics or the proof-theory

External vs. Internal Consistency. Let us consider the program $P+N = \{a \leftarrow \neg b; c \leftarrow \neg a; \neg c \leftarrow c\}$. $P+N$ is *logically inconsistent*, and thus, preferential support sanctions anything at all. Nonetheless, X^2NAF does not; X^2NAF sanctions c , $\neg b$ and a , but neither b , $\neg a$ or $\neg c$. In other words, although $P+N$ is inconsistent, the set of conclusions delivered by X^2NAF is *internally consistent*.

We can generalize this observation as follows. Let us first recall the notion of admissibility, where a set of assumptions Δ is *admissible* when it is not self-defeating and does not support \bar{N} . Then we can show the following:

Theorem 8 *Any collection of assumptions computed by X^2NAF from a program $P+N$ will be admissible as long the program P is completely stable.*

Note that the result is independent of the consistency or “openness” of $P+N$. The collection of rules in P , however, must be consistent as that is a consequence of the complete stability of P .

5 Extensions

In the previous section we showed that for the class of default programs, X^2NAF computes all and only linear stable arguments. According to the proof-theory there are then two types of arguments which X^2NAF fails to compute: non-linear stable arguments, and stable covers (i.e., disjunctive arguments).

From a logical point of view, there is the temptation to throw the proof-procedure X^2NAF and compute directly with the proof-theory. However, the proof-theory is not manageable in general. The basic operation in the proof-theory, checking whether an argument Δ is protected from a conflicting argument Δ' , involves indeed two costly steps: 1) checking that the set $\Delta_B = \Delta + \Delta' - \Delta_A$ is consistent for a chosen $\Delta_A \subseteq \Delta' - \Delta$, and 2) checking for each $\neg \beta_i \in \Delta_A$, that the atom $\neg \beta_i$ is supported by Δ_B . Furthermore, a set Δ may be protected from two sets Δ' and Δ'' and still be unprotected from their union $\Delta' + \Delta''$.

X^2NAF , although it is not complete, it is much more efficient. X^2NAF exploits the constraints imposed on the class of default programs to recursively decompose the task of checking the stability of a set $\Delta_n = \{\alpha_1, \dots, \alpha_n\}$, into the simpler tasks of checking the stability of the subsets $\Delta_i = \{\alpha_1, \dots, \alpha_i\}$, $i = 1, \dots, n$, and testing a simple condition on the minimal arguments in conflict with α_n .

In addition, the linear stable arguments computed by X^2NAF are particularly meaningful: they are the arguments which can be derived *one assumption at a time* of preferential support which accept arbitrary clauses as constraints.

time. Non-linear arguments, on the other hand, require us to entertain arguments in conflict with *multiple* assumptions, while disjunctive arguments require us to entertain *multiple* (non-disjunctive) arguments.

Yet, as we will show in this section, it is possible to do better. Indeed, there is a slight variation of the procedure X²NAF which will allow us to capture some *non-linear* stable arguments as well. Moreover, even for computing linear arguments, the new procedure X³NAF will be more efficient than X²NAF, involving a less number of tests.

5.1 Non-linear Stable Arguments

As we said above, non-linear stable arguments are arguments which *cannot* be computed one assumption at a time. The following example illustrates the intuition behind those arguments.

Example 2 The scenario involves a duel between two people a and b. For simplicity, we assume that they will shoot at each other at time $t = 1$, if they that time point alive, and that if shot, they die. The program looks as follows:¹¹

```
alive(x, s(t)) ← alive(x, t), ¬ab(x, t)
dead(a, 2) ← alive(b, 1)
ab(a, 1) ← alive(b, 1)
dead(b, 2) ← alive(a, 1)
ab(b, 1) ← alive(a, 1)
alive(a, 0) ←
alive(b, 0) ←
← alive(x, t), dead(x, t)
```

In this program, both $\neg ab(a, 0)$ and $\neg ab(b, 0)$ are derivable, and so are the atoms $dead(b, 2)$ and $dead(a, 2)$ which they support. Furthermore the set $\{\neg ab(a, 0), ab(b, 0)\}$ is linearly stable.

Now, consider two new observations:

```
dead(a, 3) ←
dead(b, 3) ←
```

In this scenario, the set $\{\neg ab(a, 0), \neg ab(b, 0)\}$ remains stable *but is no longer linearly stable*. The reason is that the observation $dead(a, 3)$ introduces a new assumption set $\Delta' = \{\neg ab(a, 1), \neg ab(a, 2)\}$ which rebuts $\Delta = \{\neg ab(a, 0)\}$ such that $\Delta' + \Delta$ is non-self-defeating. A similar thing occurs with the other assumption $\neg ab(b, 0)$.

Intuitively what is going on is that in order to defeat the sets which rebut $\neg ab(a, 0)$ and $\neg ab(1, 0)$ we need to consider both assumptions as a pair rather than

¹¹ As in example 2, integers n are used as abbreviation for n -th applications of the successor function s to 0.

individually. A variation of the procedure X²NAF will permit us to do precisely that.

Let us first recall some notions from the proof-theory of the previous section. From the proof-theory we know that an argument is stable if it protected from every conflicting argument and no conflicting argument is protected from it. Furthermore, Δ' can be protected from Δ *only* if Δ' , possibly borrowing some assumptions from Δ , supports the complement of some assumption in Δ . If that is not the case, all we need to do in order to discard Δ' is to show that Δ is protected from Δ' . However, since Δ' cannot support the negation of any assumption in Δ , under the assumption of complete stability, this amounts to checking that $\Delta' + \Delta$ is *self-defeating*.

The procedure below, called X³NAF, takes advantage of this property. For that reason, X³NAF operates not on lists of assumptions, but on lists of *assumption sets*. We use the notation $\Delta_{1,n} = \{\Delta_1, \Delta_2, \dots, \Delta_n\}$ to represent the list whose i -th element is the set Δ_i . We also write $\Delta_{1,i} = \Delta_{1,i-1} + \Delta_i$ to denote the list obtained from $\Delta_{1,i-1}$ by appending the element Δ_i . When the structure of a list $\Delta_{1,i}$ is not important, we take $\Delta_{1,i}$ to denote the collection of assumptions it contains. The definition of X³NAF proceeds as follows:

Definition (X³NAF)

1. $\Delta_{1,0} = \emptyset$ is derivable from P+N
2. $\Delta_{1,i} = \Delta_{1,i-1} + \Delta_i$ is derivable from P+N if
 - 2.1 Δ_{i-1} is derivable from P+N, and \forall admissible Δ'
 - 2.2.a if Δ' supports β_j^i for some $\neg \beta_j^i$ in Δ_i , then $\Delta' + \Delta_{1,i-1}$ is self-defeating, and
 - 2.2.b if $\Delta' + \Delta_i$ supports \overline{N} , $\Delta' + \Delta_{1,i}$ is self-defeating

The difference between X³NAF and X²NAF arises when there is an argument Δ' which is in conflict with $\Delta_i = \{\neg \beta_1^i, \dots, \neg \beta_{i_n}^i\}$ but does not support any of the atoms β_j^i . In that case, X²NAF requires that *each* of the sets $\Delta' + \Delta_{1,i-1} + \{\neg \beta_1^i, \dots, \neg \beta_j^i\}$, $j = 1, \dots, i_n$, be self-defeating, while X³NAF only requires the complete set $\Delta' + \Delta_{1,i-1} + \Delta_i$ to be self-defeating. For that reason, X³NAF is normally more efficient than X²NAF requiring a less number of tests.

In the example above, the reader can check that the list $\Delta_1 = \{\{\neg ab(a, 0), \neg ab(b, 0)\}\}$ is now derivable.

5.2 Disjunctive Reasoning

Although the new procedure handles a larger class of stable arguments, those arguments are still limited to *conjunction* of assumptions. The proof-theory, on the other hand, can handle *disjunctive* arguments as well. For example, in the program $P = \{c \leftarrow a ; c \leftarrow b ; a \leftarrow \neg b ; b \leftarrow \neg a\}$, the collection of sets $\Delta_1 = \{\neg a\}$ and $\Delta_2 = \{\neg b\}$ constitute a stable cover, and

since, each supports the atom c , c is sanctioned. The atom c , however, is not derivable by X^3NAF .

Here we face the same problem we faced before with non-linear arguments: how to achieve a good trade-off between computational and completeness concerns. In the rest of this section, we describe two simple, sound meta-rules which extend the power of X^3NAF , although they do not yield a complete characterization.

With no loss of generality we will be working with programs P with no constraints. We will use the notation $P \vdash p$ to express that an atom p is supported in all preferred models of P . From the results in the previous section we know that $P \vdash p$ will be true if P is a default program and p is obtained from P by any of the procedures X^iNAF .

We will also need to refer to the program $P[a]$ which results from the ground instantiation of P by removing all the rules with the negative literal $\neg a$ in their bodies. Similarly, the program $P[\bar{a}]$ will refer to the program obtained from the ground instantiation of P by removing all the rules with the positive literal a in their bodies.

The rules below are sound rules about the operator ' \vdash '. They can be thought as valid forms of reasoning by cases.¹²

1. If $P[a] \vdash p$ and $P[\bar{a}] \vdash p$, then $P \vdash p$
2. If $\Delta = \{\neg a_1, \dots, \neg a_n\}$ is inadmissible and $P[a_i] \vdash p$ for $i = 1, \dots, n$, then $P \vdash p$

Any of the two rules above suffices to handle the example at the beginning of this subsection. First, the set $\{\neg a, \neg b\}$ is self-defeating and thus inadmissible. Furthermore, if we exclude the rules with $\neg a$ in their bodies, the assumption $\neg b$ is sanctioned, and hence the atom c . Similarly, if we exclude the rules with $\neg b$ in their bodies, c is sanctioned as well. Thus, rule 2 allows to conclude c from P . By a slightly different argument we can also apply rule 1 and reach the same conclusion.

6 Related Work

The work in this paper is an elaboration of ideas in (Geffner, 1989, Geffner, 1990) on causal theories. Causal theories are non-monotonic theories which embed an operator C whose intuitive meaning when applied to an atom p , is that p is supported (or explained). While in preferential support, we speak of an atom p being true in a model and distinguish that

¹²Note that the logically valid rule "if $P + \{a \leftarrow\} \vdash p$ and $P + \{\leftarrow a\} \vdash p$, then $P \vdash p$ " is not valid in the context of preferential support. The reason is that such rule presumes that every atom is either false or *supported*, which is not true in general.

from p being *supported*, in causal theories we speak of p being true in a class of models as opposed to Cp being true. The semantics of *default* programs $P+N$ according to preferential support is equivalent to the semantics that results from translating each rule $\gamma \leftarrow \alpha_1, \dots, \alpha_n, \neg \beta_1, \dots, \neg \beta_m$ into causal rules of the form $C\alpha_1 \wedge \dots \wedge C\alpha_n \wedge \neg \beta_1 \wedge \dots \wedge \neg \beta_m \Rightarrow C\gamma$. Nonetheless, the framework developed in this paper is simpler than causal theories, and although it is less general, it may have all the ingredients that we need for default reasoning.¹³

As discussed throughout the paper, preferential support is also closely related to the work on stable models. Indeed, the notion of support which is central in our approach is derived from a program transformation used by Gelfond and Lifschitz. Nonetheless, the behavior of the two semantics often differs. First, our semantics applies to any program and not only programs with stable models, and then, it has a constructive proof-theory.

Regarding the scope of the stable models semantics, some extensions for handling arbitrary programs have been recently proposed (Kakas and Mancarella, 1990, Giordano and Martelli, 1990) (see also (Baral and Subrahmanian, 1990)). The idea, related to an earlier proposal by (Morris, 1989), is to augment a program P with a set of rules $p_i \leftarrow$ such that the resulting program has a stable model. These semantics have been used to characterize abductive forms of reasoning and truth-maintenance systems. Similar ideas underlie proof-theoretic approaches in which negative literals are replaced by hypothetical assumptions which can be adopted when consistent. However, in both cases, unless special care is taken, the preferences among assumptions implicit in the negation as failure rule are lost (see (Eshghi and Kowalski, 1989)).

The relation to the well-founded semantics was pointed out in section 2 where we showed that the atoms computed by X^1NAF are exactly those which are true in the (possibly partial) well-founded model of the program. One of the limitations of X^1NAF , which translates to the well-founded semantics, is the inability to handle negative information.

An extension of logic programs which accommodates classical negation has been recently advanced by Gelfond and Lifschitz (1990). However, although they accommodate classical negation they give up the classical interpretation of rules. Thus a program like $\{a \leftarrow \neg b ; \neg a\}$ is declared to be inconsistent. The applicability of that framework to problems in default reasoning appears thus limited, as we don't only want

¹³For more on this theme, see (Geffner, 1991). Let us add that the definition of preferred models in terms of bases of models is an improvement over the definitions used in causal theories. That improvement, however, can be easily carried over.

to jump to conclusions, but accept that some of those conclusions may be false.

The language of propositional logic programs we have considered is equivalent to the language of Doyle's (1979) TMSs, where rules are called justifications, negative constraints are called nogoods, and negative literals play the role of OUT-justifiers. This relation has been recently studied in (Reinfrank *et al.*, 1989, Elkan, 1988, Giordano and Martelli, 1990). Still, a great distance remains between the *procedures* used in logic programs from those used in non-monotonic TMSs (e.g., (Dressler, 1988, Junker, 1989)). Appealing to the notions introduced in section 4, it could be said that while the former aim to compute the *conclusions* supported by *linear* (decomposable) stable arguments, the latter compute 'admissible' labelings, whether linear or not. The procedures in this paper, specially X³NAF, although belong to first group, may also be relevant to non-monotonic TMSs.

7 Conclusions

We have presented a framework for logic programs with negation structured in three parts: a model-theory, a sound and complete proof-theory, and various proof-procedures. The model-theory is determined by a model-preference criterion which distinguishes atoms being true from atoms being supported. The proof-theory is based on the notion of arguments which can conflict with each other and be *protected* from each other. Finally, the proof-procedures aim to provide an efficient although incomplete approximation of the proof-theory. The resulting framework, which also accommodates explicit negative information, appears to be particularly suitable for representing and reasoning with defaults.

Acknowledgements

I want to thank Kit Fine who got me interested in negation as failure and who once explained to me the idea of using bases of models (section 4). I also want to thank Chitta Baral, Benjamin Grosof, Jorge Lobo and V.S. Subrahmanian for useful discussions.

References

- K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, Los Altos, CA, 1987.
- C. Baral and V.S. Subrahmanian. Stable and extension class theory for logic programs and default logic. *Proceedings of the Third International Workshop on Non-Monotonic Reasoning*, pages 32–43, South Lake Tahoe, CA., 1990.
- K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.
- J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.
- O. Dressler. Extending the basic ATMS. *Proceedings ECAI*, pages 385–391, 1988.
- C. Elkan. A rational reconstruction of nonmonotonic TMSs. Technical report, Cornell University, 1988.
- K. Eshghi and R. Kowalski. Abduction compared with negation as failure. *Proceedings of the Sixth International Conference on Logic Programming*, Cambridge, Mass., 1989. MIT Press.
- K. Fine. The justification of negation as failure. In J. E. Fenstad *et al*, editor, *Logic, Methodology and Philosophy of Science*, pages 263–301. Elsevier Science Publishers, 1989.
- H. Geffner. *Default Reasoning: Causal and Conditional Theories*. PhD thesis, Computer Science Department, UCLA, Los Angeles, CA, 1989.
- H. Geffner. Causal theories for nonmonotonic reasoning. *Proceedings AAAI-90*, pages 524–530, Boston, MA, 1990.
- H. Geffner. Reasoning with defaults in a logic programming framework. Technical report, IBM T.J. Watson Research Center, 1991. forthcoming.
- M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 1070–1080, Cambridge, Mass., 1988. MIT Press.
- M. Gelfond and V. Lifschitz. Logic programs with classical negation. *Proceedings of the Seventh International Conference on Logic Programming*, Cambridge, Mass., 1990. MIT Press.
- L. Giordano and A. Martelli. Generalized stable models, truth-maintenance and conflict resolution. *Proceedings of the Seventh International Conference on Logic Programming*, pages 427–441, Cambridge, Mass., 1990. MIT Press.
- S. Hanks and D. McDermott. Non-monotonic logics and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.
- C. Hewitt. Description and theoretical analysis of planner: a language for proving theorems and manipulating models in a robot. Technical Report TR-258, MIT, AI Lab., Cambridge, Mass., 1972.
- U. Junker. A correct non-monotonic ATMS. *Proceedings IJCAI-89*, pages 1049–1054, Detroit, Michigan, 1989.
- A.C. Kakas and P. Mancarella. Generalized stable models: A semantics for abduction. *Proceedings ECAI*, pages 385–391, 1990.

- R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.
- J. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, New York, 1984.
- R. Loui. Defeat among arguments: A system of defeasible inference. *Computational Intelligence*, 1987.
- P. Morris. Autoepistemic stable closures and contradiction resolution. In M. Reinfrank et al., editor, *Proceedings of the Second International Workshop on Non-Monotonic Reasoning*, pages 60–73, Berlin, Germany, 1989. Springer Lecture Notes on Computer Science.
- J. Pollock. Defeasible reasoning. *Cognitive Science*, 11:481–518, 1987.
- H. Przymusinska and T. Przymusinski. Semantic issues in deductive databases and logic programs. In A. Banerji, editor, *Sourcebook on the Formal Approaches in Artificial Intelligence*. North Holland, Amsterdam, 1989.
- T. Przymusinski. On the declarative and procedural semantics of logic programs. Technical report, University of Texas at El Paso, 1987.
- M. Reinfrank, O. Dressler, and G. Brewka. On the relation between truth maintenance and autoepistemic logic. *Proceedings IJCAI-89*, pages 1206–1212, Detroit, Michigan, 1989.
- K. Ross. A procedural semantics for well founded negation in logic programs. *Proceedings Eighth Symposium on Principles of Database Systems*, pages 22–33, 1989.
- P. Rousell. Prolog, Manuel de reference et d'utilisation. Technical report, Groupe d'Intelligence Artificielle, U.E.R. de Marseille, France, 1975.
- A. Van Gelder, K. Ross, and J. S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. *Proceedings Seventh Symposium on Principles of Database Systems*, pages 221–230, 1988.
- A. Van Gelder. The alternating fixpoint of logic programs with negation (Extended Abstract). *Proceedings Eighth Symposium on Principles of Database Systems*, pages 1–10, 1989.

Disjunctive Defaults

Michael Gelfond

Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968

Halina Przymusińska

Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968

Vladimir Lifschitz

Department of Computer Sciences
and Department of Philosophy
University of Texas at Austin
Austin, TX 78712

Miroslaw Truszczynski

Department of Computer Science
University of Kentucky
Lexington, KY 40506

Abstract

A generalization of Reiter's default logic is proposed that provides an improved treatment of default reasoning with disjunctive information. The new system — *the disjunctive default logic* — is used in the paper to reexamine the "broken-hand" example of Poole. We also compare the expressive power of this approach with two other approaches which interpret disjunctive information within the standard default logic. Finally, we show that our semantics of disjunctive default logic is a generalization of the semantics of disjunctive and extended disjunctive databases.

1 INTRODUCTION

In this paper we generalize the theory of default reasoning developed by Reiter [Rei80]. The generalization is motivated by a difficulty encountered in attempts to use defaults in the presence of disjunctive information [Poo89]. The difficulty has to do with the difference between a default theory with two extensions — one containing a sentence α , the other a sentence β — and the theory with a single extension, containing the disjunction $\alpha \vee \beta$. This difficulty was also observed by Lin and Shoham in [LS90]. They present there an example (Example 3.1 [LS90]) of a (modal) theory T , containing disjunctive information, and comment that no default theory exists that corresponds to T .

This difference may seem inessential, because the set of theorems (sentences that belong to all extensions) may be the same in both cases. But a default theory

is more than just a set of theorems. The usual explanation of the intuitive meaning of a default mentions consistency — consistency with a set of beliefs. Defaults are assertions not only about a domain described by the underlying first-order language, but also about a set of beliefs concerning that domain. A default theory with several extensions is a theory of several sets of beliefs. To formalize a body of knowledge in default logic, one has to decide what these sets of beliefs are. In particular, one needs to know whether the theory is about two sets of beliefs, or about one set containing disjunctive information. Poole's example, reproduced in Section 3 below, illustrates the crucial role of this distinction.

In the paper, we introduce an extension of Reiter's default logic by adding capabilities of handling disjunctive information. This new system — *disjunctive default logic* — is formally introduced in Section 5. We also show there how to use this new system to formalize Poole's example. Next, in Section 6, we study the expressive power of our formalization by comparing it with two systems obtained by interpreting a disjunctive default theory in the standard default logic. Finally, we show that our semantics of disjunctive default logic is a generalization of the semantics of disjunctive databases and extended disjunctive databases proposed in [GL90a]. Disjunctive default logic can be extended even further to the language of modal logic. This approach makes proofs of some of our results simpler and more elegant. We introduce this modal extension of disjunctive default logic in Section 8. Proofs of our results are gathered there, too.

2 DEFAULT THEORIES

We begin with a brief review of Reiter's default logic restricted, for simplicity, to the case of quantifier-free defaults. This restriction allows us to disregard the

process of Skolemization, involved in defining extensions in the general case; see [Rei80], Section 7.1. A default is an expression of the form

$$\frac{\alpha : \beta_1, \dots, \beta_m}{\gamma}, \quad (1)$$

where $\alpha, \beta_1, \dots, \beta_m, \gamma$ ($m \geq 0$) are quantifier-free formulas. Formula α is the prerequisite of the default, β_1, \dots, β_m are its justifications, and γ is its consequent. Informally, (1) is interpreted as follows:

$$\begin{array}{l} \text{If } \alpha \text{ holds, and} \\ \text{each of } \beta_1, \dots, \beta_m \\ \text{can consistently be assumed,} \\ \text{then infer } \gamma. \end{array} \quad (2)$$

If the prerequisite α in (1) is the formula *true*, it will be dropped; if, in addition, $m = 0$, then we identify the default (1) with its consequent γ . Thus formulas can be viewed as defaults of a special form. A default theory is a set of defaults. Let us comment here that our definition differs from the standard definition of a default theory as a pair (D, W) , where D is a set of defaults and W is a collection of (quantifier-free) formulas. As stated above, formulas can be viewed as defaults of a special form, and both approaches can easily be shown to be equivalent.

Reiter's definition of an extension shows how to make (2) precise. It can be stated as follows.

Definition 2.1 Let D be a default theory, and let E be a set of sentences. E is an extension for D if it coincides with the smallest deductively closed set of sentences E' satisfying the condition: for any ground instance (1) of any default from D , if $\alpha \in E'$ and $\neg\beta_1, \dots, \neg\beta_m \notin E'$ then $\gamma \in E'$. A theorem of a default theory is a sentence that belongs to each of its extensions.

Notice that the definition of an extension treats a default with variables as shorthand for the set of its ground instances¹.

It is often convenient to think about this definition of extensions in terms of the following two-step procedure. Let D be a default theory. The first step, which turns out to be responsible for nonmonotonicity, is to assume a hypothetical belief set E and to preprocess the set of defaults in D with respect to E .

Definition 2.2 Let D be a default theory and let E be a set of sentences. The reduct of D with respect to E , denoted D^E , is the set of inference rules defined as follows: An inference rule

$$\frac{\alpha}{\gamma}$$

¹A different approach to default logic, treating variables in defaults as “real” variables and not only as symbols for objects explicitly appearing in the theory was proposed in [Lif90].

is in D^E if for some β_1, \dots, β_m such that for every i , $1 \leq i \leq m$, $\neg\beta_i \notin E$, the default

$$\frac{\alpha : \beta_1, \dots, \beta_m}{\gamma}$$

is in D .

The second step is to define the set of theorems of the formal system obtained by expanding the system of propositional logic by the rules in D^E . This set of theorems can be defined as the set of all formulas having proofs in this system or, equivalently, as the smallest theory E' closed under provability in propositional calculus and under the rules from D^E , where E' is closed under the rule

$$\frac{\alpha}{\gamma}$$

if whenever $\alpha \in E'$, $\gamma \in E'$, as well. If this set of theorems coincides with E , E is an extension, and conversely. Formally, we have the following theorem

Theorem 2.3 A set of sentences E is an extension for a default theory D if and only if E is the minimal set E' closed under provability in propositional calculus and under the rules from D^E .

3 POOLE'S EXAMPLE

The following example from [Poo89] illustrates a difficulty that arises in some attempts to use Reiter's formalism in the presence of disjunctive information.

By default, people's left arms are usable, but a person with a broken left arm is an exception, and similarly for the right arms. One way to express this in Reiter's notation is to use “semi-normal” defaults, in the spirit of [RC81]. Below, *lh* and *rh* stand for left hand and right hand, respectively.

$$\frac{: lh\text{-usable} \wedge \neg lh\text{-broken}}{lh\text{-usable}}, \quad \frac{: rh\text{-usable} \wedge \neg rh\text{-broken}}{rh\text{-usable}}. \quad (3)$$

If you have no additional information about my arms, you will conclude that they are usable. Indeed, the default theory (3) has a single extension, containing *lh*-usable and *rh*-usable. If you know that my left arm is broken,

$$lh\text{-broken} \quad (4)$$

then this conclusion is blocked — the extension for the default theory (3), (4) contains *rh*-usable, but not *lh*-usable. So far Reiter's logic works fine.

But suppose you remember seeing me with a broken arm, you are not sure which one:

$$lh\text{-broken} \vee rh\text{-broken}. \quad (5)$$

The default theory (3), (5) has a single extension also. In addition to (5), this extension contains, unfortunately, both *lh*-usable and *rh*-usable, contrary to what we would expect.

There are other natural formalizations of Poole's example in standard default logic. One such formalization, using new variables ab_1 and ab_2 to denote abnormality, consists of the formula (5), two formulas (6) and two defaults (7).

$$lh\text{-broken} \Rightarrow ab_1, \quad rh\text{-broken} \Rightarrow ab_2 \quad (6)$$

$$\frac{: \neg ab_1}{lh\text{-usable}}, \quad \frac{: \neg ab_2}{rh\text{-usable}}. \quad (7)$$

Also this default theory has a unique extension containing both *lh*-usable and *rh*-usable, which does not agree with the intuition.

4 DISCUSSION

Let us apply the informal interpretation (2) to Poole's example. The first of the defaults (3) says: If it is consistent to assume $lh\text{-usable} \wedge \neg lh\text{-broken}$, then infer *lh*-usable. This mode of reasoning is acceptable when all available information about *lh*-broken is "vivid" — expressed by one of the literals *lh*-broken, \neg *lh*-broken (see [Lev86]). The disjunction (5) suggests *lh*-broken only as a possibility. This does not make the opposite assumption inconsistent, although we do not want the default to be applicable when this disjunction is postulated. Consistency in the defaults (2) is meant to be consistency with a set of beliefs which is not expected to contain disjunctive information. This explains why the defaults (3) lead, in the presence of (5), to unintended conclusions.

The way out is to express the incompleteness of information by the multiplicity of extensions, rather than by a single extension containing a disjunction. What we want in this example is two extensions: one containing *lh*-broken, the other *rh*-broken. Such a formalization would be similar to the default theory (3), (5), in that the disjunction (5) would be among its theorems. The difference is that each extension would be the deductive closure of a set of literals.

In the new formalism, the postulate (5) will be replaced by the expression

$$lh\text{-broken} | rh\text{-broken}. \quad (8)$$

Semantically, the difference between (5) and (8) is that the latter requires an extension to contain one of the two disjunctive terms, rather than the disjunction. This is similar to the difference between the assertions: " $\alpha \vee \beta$ is known" and " α is known or β is known."

5 DISJUNCTIVE DEFAULT THEORIES

In this section we introduce the main concepts of the paper — a disjunctive default, a disjunctive default theory. Most importantly, we extend Reiter's definition of extension to the case of disjunctive defaults.

Next, we use our approach to analyze Poole's example.

A *disjunctive default* is an expression of the form

$$\frac{\alpha : \beta_1, \dots, \beta_m}{\gamma_1 | \dots | \gamma_n}, \quad (9)$$

where $\alpha, \beta_1, \dots, \beta_m, \gamma_1, \dots, \gamma_n$ ($m, n \geq 0$) are quantifier-free formulas. Formula α is the *prerequisite* of the default, β_1, \dots, β_m are its *justifications*, and $\gamma_1, \dots, \gamma_n$ are its *consequents*. If the prerequisite α in (1) is the formula *true*, it will be dropped; if, in addition, $m = 0$, then we write the default (1) as $\gamma_1 | \dots | \gamma_n$. A *disjunctive default theory* (a *ddt*, for short) is a set of disjunctive defaults.

Definition 5.1 Let D be a disjunctive default theory, and let E be a set of sentences. E is an *extension* for D if it is one of the minimal deductively closed sets of sentences E' satisfying the condition: For any ground instance (9) of any default from D , if $\alpha \in E'$ and $\neg \beta_1, \dots, \neg \beta_m \notin E$ then, for some i ($1 \leq i \leq n$), $\gamma_i \in E'$. A *theorem* is a sentence that belongs to all extensions.

Observe that in the definition of an extension, disjunctive defaults are treated as shorthand for the sets of their ground instances.

It is clear that for standard (nondisjunctive) default theories this definition gives extensions in the usual sense, as defined in Section 2.

The definition of an extension for a *ddt* can also be described by means of the concept of *reduct*. To this end, we need more terminology. A *disjunctive rule* is an expression of the form

$$\frac{\alpha}{\gamma_1 | \dots | \gamma_n}. \quad (10)$$

We say that a theory E is *closed under a disjunctive rule* (10) if, whenever $\alpha \in E$, then there exist i , $1 \leq i \leq n$, $\gamma_i \in E$.

Remark Note that given a set S of disjunctive rules there may be several minimal sets of sentences closed under S , unlike in the case of sets of standard inference rules where there always exists the least set of sentences closed under them.

Definition 5.2 Let D be a *ddt* and let E be a set of sentences. The *reduct* of D with respect to E , denoted D^E , is the set of inference rules defined as follows: An inference rule

$$\frac{\alpha}{\gamma_1 | \dots | \gamma_n}$$

is in D^E if for some β_1, \dots, β_m such that $\neg \beta_i \notin E$, $1 \leq i \leq m$, the default

$$\frac{\alpha : \beta_1, \dots, \beta_m}{\gamma_1 | \dots | \gamma_n}$$

is in D .

We have the following straightforward theorem.

Theorem 5.3 *A set of sentences E is an extension for a ddt D if and only if E is a minimal set closed under propositional consequence and under the rules from D^E.*

The difference between disjunctive defaults and standard default with disjunctive consequents can be illustrated by the following example from [Rei80]. The default theory

$$\left\{ \frac{a : b}{b}, \frac{c : d}{d}, a \vee c \right\},$$

where a, b, c, d are propositional symbols, has a single extension, consisting of the disjunction $a \vee c$ and its logical consequences; the first two defaults “don’t work.” On the contrary, the disjunctive default theory

$$\left\{ \frac{a : b}{b}, \frac{c : d}{d}, a | c \right\} \quad (11)$$

(where $a | b$ is to be understood as a disjunctive default according to the convention introduced at the beginning of the section) has two extensions: the deductive closure of $\{a, b\}$ and the deductive closure of $\{c, d\}$. The formula $b \vee d$ belongs to both extensions, and consequently is a theorem.

The new formalization of Poole’s example is the disjunctive default theory (3), (8). It has two extensions. Each of the extensions contains one of the atoms *lh*-*usable*, *rh*-*usable* and does not contain the other. Thus neither atom is a theorem, and Poole’s paradox is eliminated.

It is also easy to see that disjunctive default logic is exactly what is needed to properly represent the theory of Example 3.1 of [LS90].

6 EXPRESSING DISJUNCTIVE INFORMATION BY STANDARD DEFAULTS

As we mentioned earlier, defaults are assertions not only about a domain described by the underlying first-order language, but also about possible sets of beliefs concerning that domain. In other words, we are interested not only in the theorems of a ddt D, but also in what sets of beliefs can be associated with D. The class of extensions for a ddt D, as defined in the previous section, can be viewed as the sets of beliefs “grounded” in the default theory D.

An important question is: Can the behavior of disjunctive defaults be simulated in standard default logic? More precisely, is there a general transformation that assigns to a ddt D a standard default theory D' (or a family D of standard default theories) so that extensions for D are exactly extensions for D' (or for theories in D).

We will consider two naturally arising transformations. The first of them “breaks” a disjunctive default

$$\frac{\alpha : \beta_1, \dots, \beta_n}{\gamma_1 | \dots | \gamma_n}$$

into n standard defaults

$$\begin{aligned} &\frac{\alpha : \beta_1, \dots, \beta_m, \neg\gamma_2, \dots, \neg\gamma_n}{\gamma_1}, \\ &\dots \\ &\frac{\alpha : \beta_1, \dots, \beta_m, \neg\gamma_1, \dots, \neg\gamma_{n-1}}{\gamma_n}. \end{aligned}$$

Given a ddt D, by D' we denote the default theory obtained by applying this transformation to every default in D. For example, for

$$D = \left\{ \frac{: b}{a | b}, \frac{: a}{a | b} \right\}$$

we have

$$D' = \left\{ \frac{: b, \neg b}{a}, \frac{: b, \neg a}{b}, \frac{: a, \neg b}{a}, \frac{: a, \neg a}{b} \right\}.$$

It is easy to see that D and D' have the same extensions. If by Cn(·) we denote the operator of propositional consequence, then these extensions are Cn({a}) and Cn({b}).

But in general, this is not the case. Consider the following theory:

$$D = \{a \Leftrightarrow b, a | b\}.$$

Then,

$$D' = \{a \Leftrightarrow b, \frac{: \neg a}{b}, \frac{: \neg b}{a}\}.$$

The theory Cn({a, b}) is the unique extension for D but D' has no extensions. Thus, the classes of extensions for D and D' are not, in general, identical. However, the following weaker property holds.

Theorem 6.1 *If E is an extension for D' then E is an extension for D.*

Remark: This theorem implies that an interpretation of a ddt D which associates with it the class of extensions for D' has “more” theorems than the logic of disjunctive defaults with extensions as defined in Section 5.

There is yet another natural way of looking at a disjunctive default theory in the standard default logic. A ddt D can be viewed as a “disjunction” of a collection of standard default theories. Namely, the family D of all standard default theories that can be obtained from D by dropping, in each of its defaults, all but one of the consequents. Every such standard default theory is called a *cover* of D. Take now all extensions for all covers of D. Will we get the set of all extensions for D?

In many cases, yes. For instance, this procedure will replace the *ddt* given by (11) by two standard default theories:

$$\left\{ \frac{a : b}{b}, \frac{c : d}{d}, a \right\}$$

and

$$\left\{ \frac{a : b}{b}, \frac{c : d}{d}, c \right\}.$$

Theory $Cn(\{a, b\})$ is the unique extension for the first theory and $Cn(\{c, d\})$ is the unique extension for the second one. This collection coincides with the collection of extensions for the theory given by (11).

But in general, this equality does not hold. Consider the disjunctive default theory

$$\{a | b, \frac{a :}{b}, \frac{: \neg a}{c}\}. \quad (12)$$

The theory $Cn(\{b, c\})$ is an extension for one of the corresponding covers:

$$\{b, \frac{a :}{b}, \frac{: \neg a}{c}\}.$$

In addition, no smaller set of sentences closed under propositional consequence is an extension for any cover of the theory (12). Yet, theory $Cn(\{b, c\})$ is not an extension for (12). Nevertheless, we have the following weaker result.

Theorem 6.2 *If E is an extension for D , then E is a minimal (with respect to inclusion) element in the class of all extensions for the covers of D .*

Remark Thus, if we associate with a *ddt* D the collection of all minimal (with respect to inclusion) elements in the class of all extensions for the covers of D , the resulting system is “more secure” (has less theorems) than the system based on the class of extensions for D , as defined in Section 5.

There is however an important class of disjunctive default theories for which both systems are equivalent.

Theorem 6.3 *If a *ddt* D consists only of justification-free defaults, then the classes of extensions for D and of minimal (with respect to inclusion) extensions for covers of D coincide.*

7 RELATION TO DISJUNCTIVE DATABASES

Disjunctive default logic with extensions as defined in Section 5 can be viewed as a generalization of the semantics for disjunctive databases proposed in [GL90a]. An extended disjunctive database is a set of rules of the form

$$c_1 | \dots | c_n \leftarrow a_1, \dots, a_k, \text{not } b_1, \dots, \text{not } b_m, \quad (13)$$

where $n, m, k \geq 0$, and a_i, b_i and c_i are literals. (The word “extended” points to the fact that the literals

can be negative.) The semantics of such databases is a generalization of the “answer set” semantics for logic programs with classical negation defined in [GL90b].

Definition 7.1 *Let P be an extended disjunctive database, and let M be a set of literals. Set M is an answer set for P if it is one of the minimal sets of literals M' satisfying the conditions:*

1. *for any rule (13) in P , if $a_i \in M'$, $1 \leq i \leq k$, and $b_i \notin M$, $1 \leq i \leq m$, then, for some i , $1 \leq i \leq n$, $c_i \in M'$;*
2. *if for some atom a ,*

$$a, \neg a \in M',$$

then M' contains all literals.

An extended disjunctive database P can be associated with a *ddt* $\text{emb}(P)$ obtained from P by replacing each rule r with its disjunctive default interpretation $\text{emb}(r)$, where for a rule r given by (13) we have

$$\text{emb}(r) = \frac{a_1 \wedge \dots \wedge a_k : \neg b_1, \dots, \neg b_m}{c_1 | \dots | c_n}.$$

We have the following theorem. It generalizes similar theorems on embedding logic programs and logic programs with classical negation in default logic [BF88, GL90b].

Theorem 7.2 *Let P be an extended disjunctive database. A set of literals M is an answer set of P if and only if M is the set of all literals from an extension for $\text{emb}(P)$.*

8 PROOFS

Our definition of an extension for a *ddt*, as well as Reiter’s definition of an extension for a (standard) default theory, treat defaults as shorthand for the set of their ground instances. Ground instances of defaults are variable-free and, thus, without loss of generality we will restrict in the proofs to the case when defaults are built of formulas of fixed propositional language \mathcal{L} .

Proofs of Theorems 2.3 and 5.3 are straightforward and are omitted.

Theorem 6.1 *If E is an extension for D' then E is an extension for D .*

Proof: Clearly, E is an extension for D'^E . In particular, E is closed under propositional consequence operator. In addition, E is closed under all rules in D'^E . Consider now a rule

$$\frac{\alpha}{\gamma_1 | \dots | \gamma_n} \quad (14)$$

from D^E . If E contains at least one γ_i , then E is closed under the rule (14). If E does not contain any γ_i , then rules

$$\frac{\alpha}{\gamma_1}, \dots, \frac{\alpha}{\gamma_n}$$

are all in D'^E . Since E is closed under the rules from D'^E , it follows that $\alpha \notin E$. Thus, E is closed under all rules in D^E .

Consider an arbitrary theory $E' \subseteq E$ closed under propositional consequence and the rules in D^E . Consider a rule

$$\frac{\alpha}{\gamma}$$

from D'^E . Then, there is a default

$$\frac{\alpha : \beta_1, \dots, \beta_m}{\gamma | \gamma_1 | \dots | \gamma_n}$$

in D , such that $\neg\beta_i \notin E$, $1 \leq i \leq m$ and $\gamma_i \notin E$, for all i , $1 \leq i \leq n$. Thus, the rule

$$\frac{\alpha}{\gamma | \gamma_1 | \dots | \gamma_n}$$

is in D^E . Since E' is closed under defaults in D^E , and $\gamma_i \notin E$, for all i , $1 \leq i \leq n$, it follows that if $\alpha \in E'$ then $\gamma \in E'$. Thus, E' is closed under the rule

$$\frac{\alpha}{\gamma}$$

and, more generally, under all rules from D'^E . Since E is an extension for D' , it follows that $E = E'$. Thus, E is minimal closed under propositional consequence and rules in D^E . Consequently, E is an extension for D . \square

Before we prove Theorems 6.2 and 6.3 we will consider one more equivalent way of defining extensions for a ddt. Let us consider a modal language \mathcal{L}_L obtained by extending the language \mathcal{L} with a single modal operator L . Operator L can loosely be interpreted as “is known” or “is believed”. Consider a formula of the form

$$L\alpha \wedge \neg L\beta_1 \wedge \dots \wedge \neg L\beta_m \Rightarrow L\gamma_1 \vee \dots \vee L\gamma_n, \quad (15)$$

where α, β_i and γ_i are formulas from \mathcal{L} . Such formulas will be called *modal rules*, and collections of modal rules will be called *modal programs*. The modal rule (15) can be interpreted as:

If α is known and it is not known that β_i ($\neg\beta_i$ is consistent), $1 \leq i \leq m$, then for some i , $1 \leq i \leq n$, γ_i is known.

Under such interpretation, modal rules are similar to disjunctive defaults. Below we show how to make this similarity precise.

The key notion is the concept of a *stable theory* ([Sta80, Moo85, MT91]. A theory $T \subseteq \mathcal{L}_L$ is stable if it satisfies the following three conditions:

ST1 $T = Cn(T)$,

ST2 If $\varphi \in T$ then $L\varphi \in T$,

ST3 If $\varphi \notin T$ then $\neg L\varphi \in T$.

Stable theories capture the intuition of belief sets of an agent with full introspection capabilities, and are of fundamental importance in nonmonotonic modal formalisms.

It is well-known ([Moo85, Mar89]) that for every theory $S \subseteq \mathcal{L}$ there is a unique stable theory T such that $T \cap \mathcal{L} = Cn(S)$. We denote this unique stable theory by $St(S)$.

Stable sets have the following property, which we will refer to as the *disjunctive property*: if T is stable and formulas φ and ψ are propositional combinations of formulas of the form $L\alpha$, where $\alpha \in \mathcal{L}$, then $\varphi \vee \psi \in T$ if and only if $\varphi \in T$ or $\psi \in T$ ([HM84, MT91]).

Stable sets can be ordered by the inclusion relation applied to their modal-free parts. Precisely, if T_1 and T_2 are stable, then $T_1 \sqsubseteq T_2$ if $T_1 \cap \mathcal{L} \subseteq T_2 \cap \mathcal{L}$. The purpose of this relation is to formalize the concept of minimal knowledge about the domain of interest.

Let I be a modal program and let $T \subseteq \mathcal{L}_L$ be a stable theory. In the next definition, we specify how to apply negation by failure to remove negation from the antecedents of the rules in I .

Definition 8.1 The reduct of I with respect to T , I^T , is the modal program obtained from I by removing all rules with $\neg L\beta$ in the antecedent, if $\beta \in T$, and by removing the conjuncts $\neg L\beta$ from all other clauses.

The next definition applies the minimal-knowledge paradigm to the reduct I^T .

Definition 8.2 A stable theory T is called a modal extension for I if and only if T is a \sqsubseteq -minimal stable theory containing I^T .

It follows from our informal comments earlier that a disjunctive default

$$d = \frac{\alpha : \beta_1, \dots, \beta_m}{\gamma_1 | \dots | \gamma_n}$$

can be interpreted by a modal rule

$$emb_L(d) = L\alpha \wedge \neg L\neg\beta_1 \dots \neg L\neg\beta_m \Rightarrow L\gamma_1 \vee \dots \vee L\gamma_n.$$

For a ddt D , by $emb_L(D)$ we denote the modal program obtained by replacing each default $d \in D$ with the rule $emb_L(d)$. We have the following theorem.

Theorem 8.3 Let D be a ddt and let $S \subseteq \mathcal{L}$ be closed under propositional consequence. Theory S is an extension for D if and only if $St(S)$ is a modal extension for $emb_L(D)$.

Proof: First notice that a rule

$$L\alpha \Rightarrow L\gamma_1 \vee \dots \vee L\gamma_n$$

is in the reduct $\text{emb}_L(D)^{\text{St}(S)}$ if and only if the rule

$$\frac{\alpha}{\gamma_1 | \dots | \gamma_n}$$

is in the reduct D^S .

Next, observe that for any theory $S' \subseteq \mathcal{L}$ that is closed under propositional consequence,

$$\text{emb}_L(D)^{\text{St}(S)} \subseteq \text{St}(S')$$

if and only if S' is closed under the rules in D^S . Indeed, by the disjunctive property,

$$(L\alpha \Rightarrow L\gamma_1 \vee \dots \vee L\gamma_n) \in \text{St}(S')$$

if and only if $\alpha \notin S'$, or at least one γ_i is in S' . Thus, the assertion follows by Theorem 5.3 and by the definition of a modal extension. \square

Now we are ready to prove Theorems 6.2 and 6.3.

Theorem 6.2. *If E is an extension for D , then E is a minimal (with respect to inclusion) element in the class of all extensions for the covers of D .*

Proof: In the same way in which we defined a cover for a ddt D , we can define a *cover* of a modal program. By Theorem 8.3, it follows that to prove Theorem 6.2, it suffices to show that if a stable theory is a modal extension for a modal program I , then T is a modal extension for a cover of I .

First, assume that C_i , $1 \leq i \leq p$, are all the covers of I . It is easy to see that I is propositionally equivalent to the disjunction

$$\Phi_I = \Phi_1 \vee \dots \vee \Phi_p,$$

where each Φ_i is a conjunction of formulas of the cover C_i . Each formula Φ_i is a propositional combination of formulas of the form $L\alpha$, where $\alpha \in \mathcal{L}$. Thus, the disjunctive property applies and if T is stable then $\Phi_I \in T$ if and only if $\Phi_i \in T$, for some i , $1 \leq i \leq p$.

Let us also observe that the reducts C_i^T , $1 \leq i \leq p$, are exactly the covers of the reduct I^T . It follows from the previous discussion that if U is stable, then $I^T \subseteq U$ if and only if $C_i^T \subseteq U$, for some i , $1 \leq i \leq p$.

Consider a modal extension T for the program I . Theory T is a \sqsubseteq -minimal stable theory containing I^T . Consequently, for some i , $C_i^T \subseteq T$. Let $U \sqsubseteq T$ be stable and such that $C_i^T \subseteq U$. Then $I^T \subseteq U$. Since T is \sqsubseteq -minimal with this property, $U = T$ and T is a \sqsubseteq -minimal stable theory such that $C_i^T \subseteq T$. Hence, T is a modal extension for C_i . \square

Theorem 6.3. *If a ddt D consists only of justification-free defaults, then the classes of extensions for D and of minimal (with respect to inclusion) extensions for covers of D coincide.*

Proof: By Theorem 8.3 it follows that in order to prove Theorem 6.3, it suffices to show the following:

If a modal program I consists entirely of the rules without conjuncts $\neg L\beta$ in the antecedent, then the classes of modal extensions for I and of \sqsubseteq -minimal extensions for covers of I coincide.

To this end, assume that C_1, \dots, C_p are all the covers of I . Observe that for any stable theory T , $I^T = I$ and $C_i^T = C_i$, $1 \leq i \leq p$. By Theorem 6.2, it suffices to prove only that a \sqsubseteq -minimal extension T for a cover of I is an extension for I . Clearly, $I^T = I \subseteq T$. Consider a stable theory $U \sqsubseteq T$ such that $I^T = I \subseteq U$. Then, as in the previous proof, we have that for some i , $C_i^T = C_i \subseteq U$. Since T is \sqsubseteq -minimal among all modal extensions for covers of I , it follows that $T = U$. Consequently, T is a modal extension for I . \square

Theorem 7.2. *Let P be an extended disjunctive database. A set of literals M is an answer set of P if and only if M is the set of literals of an extension for $\text{emb}(P)$.*

Proof: Assume that a set of literals M is an answer set for P . Put $E = \text{Cn}(M)$. Consider an arbitrary default

$$\frac{a_1 \wedge \dots \wedge a_k : \neg b_1, \dots, \neg b_m}{c_1 | \dots | c_n}$$

from $\text{emb}(P)$. Suppose that $a_1 \wedge \dots \wedge a_k \in E$, and $\neg b_i \notin E$, $1 \leq i \leq m$. Then, it follows that $a_i \in M$, $1 \leq i \leq k$, and $b_i \notin M$. Since M is an answer set for P , for some i , $1 \leq i \leq n$, $c_i \in M$. Hence, $c_i \in E$.

Let $E' \subseteq E$ be a theory closed under propositional consequence and such that for every default

$$\frac{a_1 \wedge \dots \wedge a_k : \neg b_1, \dots, \neg b_m}{c_1 | \dots | c_n}$$

in $\text{emb}(P)$, if $a_i \in E'$, $1 \leq i \leq k$, and $\neg b_i \notin E$, $1 \leq i \leq m$, then for some i , $1 \leq i \leq n$, $c_i \in E'$. Let M' be the set of literals in E' . Since E' is closed under propositional calculus, the condition 2 of Definition 7.1 holds. Consider now a rule

$$c_1 | \dots | c_n \leftarrow a_1, \dots, a_k, \text{not } b_1, \dots, \text{not } b_m$$

from P . Assume that $a_i \in M'$, $1 \leq i \leq k$, and $b_i \notin M$, $1 \leq i \leq m$. Then, $a_i \in E'$, $1 \leq i \leq k$, and $\neg b_i \notin E$, $1 \leq i \leq m$. Thus, for some i , $1 \leq i \leq n$, $c_i \in E'$. Consequently, $c_i \in M'$ and M' satisfies condition 1 of Definition 7.1. Since M is a minimal set satisfying this condition and $M' \subseteq M$, $M = M'$ follows. Thus, $E = E'$ follows, which implies that E is an extension for $\text{emb}(P)$.

The converse implication can be proved in the same manner. \square

Acknowledgments

We are grateful to Fangzhen Lin for his comments on an earlier draft of the paper. The first and third authors acknowledge the support of the NSF grant IRI-89-06516. The fourth author acknowledges the support of the Army Research Office under grant DAAL03-89-K-0124, and of National Science Foundation and the Commonwealth of Kentucky EPSCoR program under grant RII 8610671. The paper was written when the second author worked at Stanford University and was supported in part by NSF grant IRI-89-04611 and by DARPA under Contract N00039-84-C-0211.h

References

- [BF88] N. Bidoit and Ch. Froixdevaux. Negation by default and non stratifiable logic programs. Internal report, 437, 1988.
- [GL90a] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. Submitted for publication, 1990.
- [GL90b] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. Warren and P. Szeredi, editors, *Logic Programming: Proceedings of the 7th International Conference*, pages 579–597, Cambridge, MA., 1990. MIT Press.
- [HM84] J.Y. Halpern and Y. Moses. Towards a theory of knowledge: preliminary report. In R. Reiter, editor, *Proceedings of the workshop on non-monotonic reasoning*, pages 125–143, 1984.
- [Lev86] H. Levesque. Making believers out of computers. *Artificial Intelligence*, 30:81–108, 1986.
- [Lif90] V. Lifschitz. On open defaults. In J. Lloyd, editor, *Computational Logic: Symposium Proceedings*, pages 80–95. Springer Verlag, 1990.
- [LS90] F. Lin and Y. Shoham. Epistemic semantics for fixed-points non-monotonic logics. In *Proceedings of TARK-90*, pages 111–120, San Mateo, CA., 1990. Morgan Kaufmann.
- [Mar89] W. Marek. Stable theories in autoepistemic logic. *Fundamenta Informaticae*, 12:243–254, 1989.
- [Moo85] R.C. Moore. Semantical considerations on non-monotonic logic. *Artificial Intelligence*, 25:75–94, 1985.
- [MT91] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 1991. To appear.
- [Poo89] D. Poole. What the lottery paradox tells us about default reasoning. In *Principles of Knowledge Representation and Reasoning*, pages 333–340, San Mateo, CA., 1989. Morgan Kaufmann.
- [RC81] R. Reiter and G. Crisculo. On interacting defaults. In *Proceedings of IJCAI-81*, pages 270–276, 1981.
- [Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
- [Sta80] R.C. Stalnaker. A note on nonmonotonic modal logic. Unpublished manuscript, 1980.

Partial Programs

Michael R. Genesereth
 Computer Science Department
 Stanford University
 Stanford, California 94305

Jane Yung-jen Hsu
 Computer Science Department
 Stanford University
 Stanford, California 94305

Abstract

Traditional programs are inappropriate for controlling the behavior of an agent in dynamic environments. This paper introduces *partial programs*, which allow one to write programs without making arbitrary design decisions and thereby facilitate incremental program development. A partial program specifies a set of constraints on the potential behavior of an agent in such a way that the agent can be informed to change its future actions. The formal syntax and semantics of a partial programming language are presented along with a sample partial program interpreter. In addition, this paper discusses and proves some important properties of partial programs.

1 INTRODUCTION

Logical specifications of knowledge about the world, manipulated by general-purpose inference algorithms, have been the focus of much research in knowledge representation and reasoning. On the other hand, procedural information has typically been specified using imperative programming languages. This paper argues for the advantages of a *logical specification of procedures*, and introduces a formalism for representing procedural knowledge so that a machine can reason about its actions and be informed, e.g. *procedural hints*, to change its future behavior.

A *program* is a description of the intended behavior of a machine encoded in some programming language. Traditional programming requires a detailed and precise specification of the machine's behavior in every state of its operation. As a result, programs in most traditional programming languages are *complete* in the sense that the behavior of the machine is completely determined by the procedure specified in the given program and its initial environment.

Due to limited sensing, acting or reasoning capabilities, a machine acting in complex environments must often deal with incomplete and dynamically changing information. The standard way to deal with incomplete information is to write conditional programs. A conditional program explicitly considers every possible values for every missing piece of information, and specifies the appropriate behavior for each case. For example, suppose we want to design a program for planning a trip from San Francisco to any other city in the United States, but the specific destination is not known ahead of time. One could write a program consisting of a big conditional statement with one branch for each potential destination city and its associated plan. In general, constructing programs with such high degree of conditionality demands a substantial amount of computation, and can sometimes be intractable. Even when it is computationally feasible to write such conditional programs, it may not be economical to do so since many of the branches will never be executed.

Furthermore, for many complex domains, it is very difficult to identify all the conditions that might occur during program execution. Writing complete programs under such circumstances can be problematic. When complete information is not available at design time, arbitrary decisions are often made prematurely in order to write runnable programs. When new information becomes available, it may be necessary to modify the agent's behavior accordingly. Subsequent modifications to a complete program involve identifying earlier design decisions and rewriting the code, which can be very time-consuming, error-prone or even prohibitively expensive.

1.1 A MOTIVATING EXAMPLE

Partial information is commonplace in many interesting domains. Let's consider a simple example in the *Square World*. The world consists of four interconnecting cells laid out on a 2-by-2 square as shown in Figure 1. There is a robot (i.e. the agent) in one of the cells and a piece of gold in another, although their

initial positions are not known in advance.

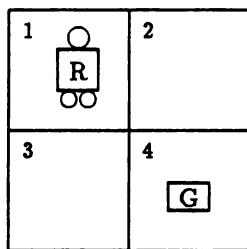


Figure 1: One State of the Square World

The robot is able to execute one of five actions: *move clockwise*, *move counterclockwise*, *grab the gold*, *drop the gold*, and *noop* (i.e. do nothing). The robot is also able to perceive whether the gold is in its own grip, in the same cell, or elsewhere; the robot cannot identify its exact location but it does recognize whether it is in the starting cell. Our task is to program the robot so that, given any initial configuration, the robot agent will find the piece of gold and then bring it back to the robot's starting position.

Figure 2 shows one possible conditional program for our Square World robot, presented in a pseudo programming language. Provided that there is exactly one piece of gold in the square and all the actions can be carried out successfully, this program is guaranteed to achieve the given goal for any initial locations of the robot and the gold. However, having committed

```

IF gold is in same cell
THEN Do Nothing
ELSE Move Clockwise
    WHILE robot is not in initial cell
        IF gold is in same cell
        THEN Grab Gold
        Move Clockwise
    END WHILE LOOP
Drop Gold

```

Figure 2: A Program for the Square World Robot

to the program in Figure 2, there is no flexibility in the agent's behavior to take advantage of new information about the world. For example, if we are given the additional information that the robot is in the upper left cell and the gold is in the lower left cell initially, then the sequence of actions shown in Figure 3 constitutes a more efficient program for the robot. The premature choice of moving clockwise rather than counterclockwise in Figure 2, in the absence of any particular justification for such a decision, made it impossible for the agent to achieve its goal by taking the alternative actions. For this simple case, the potential savings that could result from utilizing the additional information

```

Move Counterclockwise
Grab Gold
Move Clockwise
Drop Gold

```

Figure 3: An Alternative Program

may seem rather trivial. In more complicated situations (e.g. when the number of cells in the square is large), the savings can be quite significant.

A more serious problem arises when any of the implicit assumptions made originally turns out to be false. For instance, the agent may fail to carry out its intended actions, say, if the path between any two cells is blocked. Let's suppose that the robot starts out from the initial state shown in Figure 1, and it later discovers that the path between cell 3 and cell 4 has been blocked. Since the original program didn't check to see if moving clockwise is applicable, the robot will be stuck in cell 4. In fact, the robot could have achieved its goal if its program admits the sequence of actions below:

```

Move Clockwise
Move Clockwise
Grab Gold
Move Counterclockwise
Move Counterclockwise
Drop Gold

```

Figure 4: Yet Another Program

This simple example demonstrates the problems with arbitrary decisions in traditional programming. A bad decision can lead to inefficient or even unsuccessful program execution. In order to cope with incomplete information more effectively, we need a better way to program in dynamic environments.

1.2 A NEW APPROACH

This paper introduces *partial programs*, which allow one to write programs without making arbitrary design decisions and thereby facilitate the implementation of subsequent design refinements. Intuitively, a partial program specifies a set of constraints on the potential behavior of an agent, and it can be partial in the sense that it need not *uniquely* prescribe the agent's actions for *every* situation. For example, a partial program for the Square World agent may consist of the statements in Figure 5, which are written in English-like sentences before the formal syntax is introduced. Additional statements can be added incrementally to further constrain the agent's actions.

```

IF robot is in initial cell and
gold is in same cell
THEN Do Nothing

IF robot is not in initial cell and
gold is in same cell
THEN Grab Gold

IF robot is in initial cell and
gold is in grip
THEN Drop Gold

IF robot is not in initial cell and
gold is in grip
THEN do not Drop Gold

```

Figure 5: A Partial Program

In Section 2, we will present the theory of partial programs, their formal syntax and semantics, and a sample partial programming language *PPL*. Section 3 gives the description of a partial program interpreter that is shown to be sound with respect to the program specification. In particular, we shall illustrate that an agent will exhibit the desirable behavior in Figures 3 and 4 when its partial program is augmented with the additional information. Several important properties of partial programs are discussed in Section 4. Finally, we briefly review some related work that helped shape the idea of partial programs and conclude with a discussion of future research directions.

2 PARTIAL PROGRAMS – SYNTAX AND SEMANTICS

We now introduce partial programs as a way to control the behavior of a machine via a declarative specification of its intended procedure. In particular, we are interested in machines that are capable of perceiving and acting in its environment. We call such a machine an *agent* and its environment the *world*. A state is a snapshot of the world at a given point in time. An agent interacts with the world through sensors and effectors. The sensory limitations partition the states of the world into a set of *percepts* P . For example, when the Square World robot is in the upper left cell, it cannot distinguish the state in which the gold is in the upper right cell from the states in which the gold is in the lower right cell or the lower left cell. As a result, the states of the Square World are partitioned into the following six percepts (the subscript 0 indicates that the robot is in its starting position): sc_0 , sc (the gold is in the same cell with the robot), ig_0 , ig (the gold is in the grip of the robot), ew_0 , and ew (the gold is elsewhere). Let's assume that the agent perceives the

world at every time point¹ and then chooses to perform one action. The sequence of percepts from the initial time point up to now is called the perceptual history, and the set of all possible perceptual histories is denoted by P^* . The behavior of such an agent is also referred to as a *procedure*, which can be defined formally as:

Definition 1 *A deterministic procedure for an agent is a triple $\langle P, A, b \rangle$, where P and A are the percepts and actions of the agent, and b is the behavior function of the agent.*

For the presentation that follows, we further define b to be a function $b : P^* \rightarrow A$ that maps each perceptual history into an action. There are several alternative formulations of the behavior function such as $b : P \rightarrow A$ that maps the current percept into an action, $b : N \rightarrow A$ that maps the current time point into an action, and so forth. Although they differ in their expressive power, similar results hold for the various formulations.

2.1 SYNTAX

A partial program for an agent is a possibly partial specification of its procedure $\langle P, A, b \rangle$, and it consists of descriptions about its percepts and actions as well as constraints on its behavior function. Given the formulation earlier, the percepts can be viewed as the inputs to the program that are determined by the world and hardware limitations; while the actions are viewed as the outputs from the program. In some cases, we may have a description that constrains the set of actions but does not pinpoint a particular action. In other cases, the program may specify actions for some perceptual histories but not for others. Let \mathcal{L} be a language defined by the normal syntax of first order predicate calculus. The set of constant symbols C in the language are classified into objects O , functions F and relations R . Given the universe of discourse U , we can specify the percepts and actions of an agent via a naming function $\mathcal{N} : U \rightarrow C$ that enables us to refer to things in the world using symbols from \mathcal{L} . For instance, $\mathcal{L}(noop) = Noop$ and $\mathcal{L}(sc_0) = SC0$ etc².

Definition 2 *Given a programming language \mathcal{L} and a naming function \mathcal{N} for an agent with percepts P and actions A , a partial program written in \mathcal{L} is a 4-tuple $\langle \beta, \Pi, \Gamma, \Delta \rangle$, such that $\beta \in F$ is a function constant in \mathcal{L} denoting the behavior function of the agent; $\Pi \subset$*

¹In this framework, time is assumed to be linear, and it is formulated as a set N of time points.

²A note on notation: objects in the universe of discourse are written in italic; symbols in the programming language are written in typewriter font; Greek alphabets and character strings enclosed in angle brackets “⟨” and “⟩” are part of the metalanguage for talking about expressions in the language.

O and $\Gamma \subset O$ are disjoint sets of object constants as defined by \mathcal{N} , i.e. $\Pi = \{\pi \mid \mathcal{N}(p) = \pi, \text{ for some percept } p \in P\}$ and $\Gamma = \{\gamma \mid \mathcal{N}(a) = \gamma, \text{ for some action } a \in A\}$; and Δ is a database of consistent well-formed formulas in \mathcal{L} .

The function constant β is also referred to as the name of the program. The database Δ describes the intended behavior of the agent. In particular, sentences that contain the function constant β put constraints on the potential actions of the agent. For instance, sentences of the form $(\text{condition}) \Rightarrow \beta(\eta) = \gamma$ prescribes actions; while sentences of the form $(\text{condition}) \Rightarrow \beta(\eta) \neq \gamma$ prohibits actions (where η denotes a perceptual history and γ denotes an action of the agent.) A programmer may describe a procedure in as much or as little detail as he desires. It is important to realize that the only constraint on the sentences comprising a partial program is *consistency*. Among other things, an action cannot be both prescribed and prohibited for the same perceptual history; a partial program should not prescribe more than one action for any specific perceptual history; and a partial program should not prohibit *all* the actions for any specific perceptual history (which is inconsistent with the fact that the name of the program is a function symbol).

2.2 PPL

Let's look at the specifics of a sample partial programming language *PPL*. The syntax of *PPL* is based on KIF (Knowledge Interchange Format), which is an extension of first-order predicate calculus [Genesereth et al., 1990]. Every constant symbol is made up of a sequence of alphabetic or numeric characters starting with an upper case alphabetic character or a digit. Every variable in *PPL* starts with either a “\$” or a “@” and followed by a sequence of alphanumeric characters. The \$-variables are individual variables and the @-variables represent sequences. There are also some primitive functional (e.g. +, - etc.) and relational operators (e.g. =, <, > etc.) pre-defined in the language. The definitions of terms, atomic formulas and well-formed formulas follow the standard definitions for predicate calculus.

In addition to the logical specifications about the world (e.g. the connectivity between the cells, the existence of exactly one piece of gold in the world etc.), we also need expressions to describe the agent, its percepts and perceptual histories, and its actions. Therefore, several special relation constants are defined in *PPL*. A precise encoding of a partial program for our Square World robot will be presented throughout the rest of this section.

Agent specification Suppose that the object constant R32 denote the name of an agent as described in Section 1.1. The basic facts about R32 can be captured

by the following sentences.

(Agent R32)

(BehaviorFunction R32 Square-BF)

(Percepts R32 (List SC0 SC IGO IG EWO EW))

(Action R32 Move-Clockwise)

(Action R32 Move-Counterclockwise)

(Action R32 Grab-Gold)

(Action R32 Drop-Gold)

(Action R32 Hoop)

These sentences have the obvious meanings. The relation constant Percepts is an abbreviated notation of a set of facts of the form $(\text{percept } \langle \text{agent} \rangle \langle \text{percept} \rangle)$. It should be noted, however, that an object constant Square-BF is used for denoting the name of the behavior function³. This way, it is fairly straightforward to specify properties about the behavior function in first order logic, provided that the database contains axioms asserting its functional property. We will give examples of such axioms later.

History-dependent facts While some facts about the world are independent of time, many are fluents. The notations used in *PPL* resemble those of situation calculus [Kowalski, 1979b] except that terms designating situations are replaced with perceptual histories. A perceptual history is represented by a special functional term $(\text{History } \langle h \rangle \langle \text{time} \rangle)$, where $\langle \text{time} \rangle$ is a term designating a time interval, and $\langle h \rangle$ is the percepts corresponding to the specified time interval. We call such terms *history designators*. The time specification is optional, and when it is omitted, the term $\langle h \rangle$ is the sequence of percepts p_n, p_{n-1}, \dots, p_0 with p_n being the current percept and p_0 being the percept at time 0. For simplicity, no explicit mention of time is used in our example and the term $(\text{Current } \langle p \rangle)$ is used as a shorthand notation for all the terms $(\text{History } \langle h \rangle)$ such that $\langle h \rangle$ is a perceptual history with $\langle p \rangle$ as the current percept.

A history-dependent fact can thus be represented by a relation $(\text{Holds } \sigma \eta)$ where σ is a state descriptor and η is a history designator. One such fact is the following:

($\Leftarrow (\text{Holds } (\text{InGrip R32 G17}) (\text{History } \$p \$h))$
 $\quad (= \$p \text{ IG}))$

It states that if ig is the current percept, then it is true that the piece of gold (denoted by G17) is in the grip of the robot. The sentence $(\Leftarrow \phi \phi_1 \dots \phi_n)$ is true if and only if the sentence ϕ is implied by the sentences ϕ_1 through ϕ_n .

³Alternatively, the function constant Square-BF is treated as a first class object.

Action specification rules Last but not least, a partial program can describe the agent's behavior function, in other words, specifications of its actions. Since an agent's behavior function in this formulation is a mapping from perceptual histories into actions, the behavior of an agent is specified by statements relating actions with histories. There are four types of action specification rules:

- (*Applicable* $\gamma \beta \eta$)
- (*Good* $\gamma \beta \eta$)
- (*Bad* $\gamma \beta \eta$)
- (*Better* $\gamma_1 \gamma_2 \beta \eta$)

The first three specify that the action γ *may*, *should*, or *should not* be executed in the perceptual history η for the behavior function β being specified. The last one states that the agent should prefer executing action γ_1 to executing action γ_2 in η . These specifications can also be in conditional forms. As will be presented shortly, the semantics of partial programs is defined in such a way that all the actions are assumed to be applicable for every history at the start of program execution. The axiom below can be regarded to be in the database by default.

```
(<= (Applicable $a $bf $h)
      (BehaviorFunction $agent $bf)
      (Action $agent $a))
```

Some action specifications rules for the Square World robot are:

```
(Good Noop Square-BF (Current SC0))
(Good Grab-Gold Square-BF (Current SC))
(Good Drop-Gold Square-BF (Current IG0))

(Bad Drop-Gold Square-BF (Current IG))

(<= (Bad Grab-Gold Square-BF (Current $p))
     (/= $p SC) (/= $p SC0))

(<= (Bad Drop-Gold Square-BF (Current $p))
     (/= $p IG) (/= $p IG0))

(<= (Better $a Noop Square-BF $h)
     (/= $a Noop)
     (NOT (Provable(Good Noop Square-BF $h)))
     (NOT (Provable(Bad $a Square-BF $h))))
```

In a simple domain, the behavior of the agent may be dependent on its current percept only, as is the case in the program shown above. Such agents are called *Markov agents*. A Markov procedure is usually simpler to describe and more efficient to execute, but it does not necessarily produce the best sequence of actions for achieving the goals. This is because the agent will be able to make a more informed choice when more

information about its previous percepts (and actions) are taken into consideration⁴.

The functional property of the behavior function is ensured by the following axioms:

```
(<= (Bad $a2 $bf $h)
      (Good $a1 $bf $h)
      (/= $a1 $a2))

(<= (NOT (Good $a $bf $h))
      (Bad $a $bf $h))

(NOT (Forall $a (Bad $a $bf $h)))
```

In other words, if an action is prescribed, then all the other actions are prohibited; if an action is prohibited then it cannot be prescribed at the same time; a partial program cannot prohibit all the actions for any specific perceptual history.

2.3 SEMANTICS

A partial program puts constraints on an agent's choices of actions. We further assume that the agent is free to choose any one of the non-prohibited actions (subject to the constraints specified by the preference rules) whenever no action is uniquely prescribed by the program. Since the behavioral descriptions contained in a partial program only partially constrain the actions for the given agent, a typical partial program does not necessarily define a unique procedure. We define the intended meaning of a partial program to be the set of behavior functions, called its *extension*, that are compatible with the specifications in the partial program. More precisely,

Definition 3 Given a partial program $\langle \beta, \Pi, \Gamma, \Delta \rangle$ for an agent with percepts P , actions A , and a naming function N . The extension of the partial program β , written $\epsilon_N(\beta)$, is the set of behavior functions from P^* into A , such that for every behavior function $f \in \epsilon_N(\beta)$, there exists an interpretation I that satisfies the following conditions:

1. $I(\beta) = f$
2. $\models_I \Delta$
3. $\forall \pi \in \Pi \ N(I(\pi)) = \pi$
4. $\forall \gamma \in \Gamma \ N(I(\gamma)) = \gamma$

An interpretation for a language is a mapping from elements in the language into elements in the universe

⁴It is also possible to reason about possible future information (e.g. effects of actions) if modal operators are used to express expectation, possibility etc. Unlike classical planning systems such as STRIPS [Fikes and Nilsson, 1971], the effects of actions are not regarded as necessarily true in a partial programming system.

of discourse. The first condition states that under the interpretation \mathcal{I} , the function symbol β denotes a behavior function f . The second condition states that \mathcal{I} is a model of Δ , i.e. the database Δ is satisfied by the interpretation \mathcal{I} for all variable assignments. The last two conditions state that the interpretation \mathcal{I} is consistent with the given naming function \mathcal{N} . A partial program is undefined, i.e. no behavior function exists in its extension, if its database is inconsistent.

The partial program given in Section 2.2 constrains the behavior function of the Square World agent as shown below (where cw stands for moving clockwise and ccw stands for moving counterclockwise):

Table 1: Constraints on the Behavior Function.

$b : P \rightarrow A$	sc_0	sc	igo_0	ig	ew_0	ew
cw	X	X	X			
ccw	X	X	X			
$grab$	X	✓	X	X	X	X
$drop$	X	X	✓	X	X	X
$noop$	✓	X	X	△	△	△

It should be noted that this behavior function is represented by a Markov function from percepts into actions. Any entry marked with “✓” indicates that the action is prescribed for the corresponding percept; entries marked with “X” are prohibited; entries marked with “△” are less preferred than the blank entries, which indicate insufficient information for any conclusion to be drawn. The actions are uniquely prescribed for some of the percepts such as sc_0 , sc and igo_0 ; while only partially constrained for the others. The preference information can be viewed as soft constraints on the actions, or be strengthened by the following:

```
(<= (Bad $a2 $bf $h)
  (Better $a1 $a2 $bf $h)
  (/= $a1 $a2))
```

Consequently, the extension for the partial program shown above contains eight different behavior functions (two possible action choices for each of ig , ew_0 and ew) and some of them are shown below⁵:

Table 2: Behavior Functions in the Extension.

	sc_0	sc	igo_0	ig	ew_0	ew
b_1	$noop$	$grab$	$drop$	cw	cw	cw
b_2	$noop$	$grab$	$drop$	cw	cw	ccw
b_3	$noop$	$grab$	$drop$	cw	ccw	cw
...	...					

⁵In fact, one might recognise that the function b_1 corresponds to the procedure described by the conditional program given in Figure 2.

3 AN INTERPRETER

Now that we have presented the syntax and semantics of partial programs, let's look at how partial programs are executed. Even though a partial program is only a partial description of a procedure, it is possible to construct an interpreter that behaves in accordance with that description. Of course, the partial program interpreter must occasionally make arbitrary choices when no specific action is prescribed by the program. What is interesting about the partial program interpreter is that if it is given a program augmented with a constraint that prohibits this arbitrary choice, it will adjust its activity accordingly.

There are several possible architectures for executing partial programs. A brief description of a sample *compulsive deliberation* interpreter *PPI* is shown in Figure 6. This interpreter consists of a simple exit-free *perceive-deliberate-act* loop. Each time around the loop, the interpreter observes the world to identify the current percept, selects an action based on the perceptual history, and then acts on the chosen action. It reasons about the appropriate action to perform at *every* time point, and records all the perceptual information as well as the actions that have been executed. We also assume the availability of a function *Perceive()* that receives the current sensory information from the sensors and convert it into the symbol corresponding to the current percept⁶, and a function *Act(κ)* that sends the appropriate signals to the effectors of the agent so that the action $\mathcal{N}^{-1}(\kappa)$ will be carried out⁷.

Initialize perceptual history: $\eta \leftarrow ()$.

Loop Forever

1: Observe the environment: $\pi \leftarrow \text{Perceive}()$.

Update perceptual history: $\eta \leftarrow \text{update}(\eta, \pi)$.

2: Decide which action to perform:

FOR every action $\gamma \in \Gamma$, prove $\beta(\eta) \neq \gamma$.

$\Gamma' \leftarrow$ actions for which the proof has failed.

$\kappa \leftarrow \text{Choose-one}(\Gamma')$.

3: Execute the prescribed action: $\text{Act}(\kappa)$.

Record the action executed in the database Δ .

End Loop

Figure 6: A Partial Program Interpreter *PPI*

Informally, the interpreter has to choose any action that has been uniquely prescribed by the database. Otherwise, if no action is prescribed, the interpreter can choose any action that is not prohibited as long as no other non-prohibited action is preferred. The execution of the chosen action should not result in any violation of constraints in the partial program.

⁶The *signals-to-symbols* problem is by no means trivial. However, it is not the focus of this research.

⁷Function \mathcal{N}^{-1} is the inverse of the naming function \mathcal{N} .

In order to decide the right action for the given perceptual history η , the interpreter enumerates the agent's possible actions and for each action γ uses a resolution theorem prover in an attempt to decide if γ is inconsistent with the database Δ . If the proof for $\beta(\eta) \neq \gamma$ succeeds, i.e. $\Delta \cup \{\beta(\eta) = \gamma\}$ is unsatisfiable, the action is eliminated from consideration. If the inference procedure finitely fails, then action γ must be consistent with Δ . When there is more than one consistent action, the interpreter makes an arbitrary choice (which is usually the first or the "best" action found within a limited time bound).

An alternative way to implement Step 2 of the interpretation process in Figure 6 is by simplifying the term $(\text{Act } \beta \eta)$, where Act is a function defined below, which states that the value of the term $(\text{Act } \$bf \$h)$ is one of the actions that are applicable but not proven to be prohibited.

```
(<- (Act $bf $h))
  (Oneof $a (AND
    (Applicable $a $bf $h)
    (NOT (Provable (Bad $a $bf $h))))))
```

The relation $(<- \tau_1 \tau_2)$ indicates that the terms τ_1 and τ_2 refer to the same object. The variables $\$bf$ and $\$h$ will be bound to the name of the program β and the current perceptual history η respectively. The function $(\text{Oneof } \tau \phi)$ takes a term and a sentence as input and returns the term τ instantiated by the bindings of the variables in the first proof of ϕ . The value is 0 if no such proof exists. For more details, please refer to [Epikit, 1989]. The relation $(\text{Provable } \phi)$ takes a sentence as input and returns true if and only if ϕ can be proven from the set of facts in the database Δ . The database and the inference procedure used by Provable are implicit.

Furthermore, when the behavior of an agent is a Markov procedure, the fact that its actions are defined only in terms of the current percepts can be specified as the relation:

```
(<- (Act Square-BF (History $p @h))
  (Act Square-BF (Current $p))).
```

Any attempt to find the appropriate action for a given perceptual history will be simplified to finding the action for any perceptual history such that it has $\$p$ as the current percept and thus eliminating some unnecessary unifications. Moreover, we can use a specialized Markov interpreter that does not record and reason about historical information, and is much more efficient than the general-purpose interpreter.

3.1 SOUNDNESS RESULTS

One of the most important issues in interpreting a partial program is to ensure its correctness with respect

to the given specification. An arbitrary decision that turns out to be wrong in light of new information can be catastrophic; for when the database becomes inconsistent, the interpreter can prove the inconsistency of every action and ceases to act. In what follows, we will present a definition of the correctness of partial program interpretation via the notion of *faithfulness*. An *execution* of a partial program interpreter on any given partial program is a partial function from perceptual histories into actions. The function is defined in a perceptual history only if the machine running the program succeeds in finding an action to execute, in which case the value of the function is the action executed. If either the interpreter never tries to compute an action or the inference procedure never terminates, the function is undefined for that history. Intuitively, an interpreter is said to be *faithful* if given any partial program specification, its execution is compatible with some behavior function in the extension.

Definition 4 *An interpreter is faithful if given an arbitrary partial program $(\beta, \Pi, \Gamma, \Delta)$ for any agent with percepts P and actions A , the partial function g corresponding to its execution satisfies the following conditions:*

- (1) *g is a partial mapping from P^* into A .*
- (2) *$\forall \eta \in \Pi^* \forall \gamma \in \Gamma$,*
if $\Delta_\eta \models \beta(\eta) = \gamma$ then $g(\mathcal{N}^{-1}(\eta)) = \mathcal{N}^{-1}(\gamma)$.
- (3) *$\forall h \in P^* \forall a \in A$,*
if $g(h) = a$ then $\neg(\Delta_{\mathcal{N}(h)} \models \beta(\mathcal{N}(h)) \neq \mathcal{N}(a))$.

The database Δ_η denotes the database Δ augmented with the percepts and actions up to the current point (after seeing η). The first condition ensures that the execution is defined over the same percepts and actions. The second condition states that if an action is prescribed by the database, it should be executed by the interpreter. The third condition says an interpreter should never execute an action that is prohibited by the partial program.

In general, it is very difficult to guarantee the faithfulness of an interpreter. However, by restricting the partial program database to be *history-dependent* (i.e. action prescriptions depend only on past information), we can show that compulsive deliberation is faithful. History-dependency ensures that any information that can affect the decision about the agent's actions is there when the decision has to be made.

Theorem 1 *Given a sound and complete proof procedure, the compulsive deliberation interpreter PPI is faithful with respect to history-dependent partial programs.*

Proof: Let's suppose PPI uses a sound and complete proof procedure such as resolution for clauses.

If $\Delta_\eta \models \beta(\eta) = \gamma$, it follows that $\Delta_\eta \models \beta(\eta) \neq \gamma'$ for any other action γ' . Given that the theorem prover is complete, it will eventually reach the above conclusion. Therefore, γ will be the only remaining action that is not prohibited and it will be executed by the interpreter. On the other hand, if the interpreter is not faithful, there exists some $h \in P^*$ and $a \in A$ such that $g(h) = a$ while $\Delta_{N(h)} \models \beta(N(h)) \neq N(a)$. However, if $\Delta_{N(h)} \models \beta(N(h)) \neq N(a)$, the resolution theorem prover would have discovered a proof to that effect, and a would not have been executed in h . \square

The partial program interpreter *PPI* (and several variants) has been implemented in Lisp, and it uses the knowledge representation and reasoning facilities provided by a logic-based system Epikit [Epikit, 1989]. In particular, an efficient proof procedure called *paralog* was used [Sikka et al., 1991].

3.2 PROBLEMS

Since the database of a partial program is updated at every time point during the interpretation process, the contents of the database changes dynamically. One possible way to cope with this complexity is by adopting some sort of temporal logic as part of the representation language [Shoham and Goyal, 1988]. Instead, we have opted for viewing the database at each time point to be distinct. Let $\Delta(t_i)$ represent the database at time t_i . We can construct $\Delta(t_i)$ from $\Delta(t_{i-1})$ by updating the term representing the perceptual history (e.g. adding the current percept to the front of the existing history) and asserting the action executed at time t_{i-1} .

Recall that consistency is the basic requirement for the database of a partial program. Due to the dynamic nature of the partial program database, ensuring consistency can be arbitrarily hard. When the database is not history-dependent, there are cases in which the interpreter does not produce correct behavior even though the database is consistent at the beginning. Such situations arise when the action specification contains information that is not known at the time when the agent has to act, or that may later be modified by the agent's choice of actions and future percepts. Consider the program *Invest* with database Δ_1 :

```
(<= (Good Sell Invest $h)
     (Holds Price-will-fall $h))

(<= (Good Buy Invest $h)
     (Holds Price-will-rise $h))

(<= (Holds Price-will-fall $h)
     (Holds (Price $x) $h)
     (Holds (Price $y) (Next $h))
     (> $x $y))
```

```
(<= (Holds Price-will-rise $h)
     (Holds (Price $x) $h)
     (Holds (Price $y) (Next $h))
     (< $x $y))
```

The term *(Next \$h)* denotes the perceptual history at the next time point. This database specifies that if the price of stocks is going to fall, then the agent should sell stocks; otherwise if the price of stocks is going to go up, then the agent should buy more stocks. Although this may seem like a perfectly reasonable thing to say, it presents a serious problem for any agent (including human beings!). The difficulty stems from the fact that the action prescriptions for the agent depend on future information, i.e. the price of stocks at the next time point, thus the appropriate actions can be decided only after they have been taken. More specifically, suppose at time t_1 the agent makes an arbitrary decision to buy stocks. Unfortunately, the price of stocks turns out to be lower at time t_2 , and thus invalidating the earlier decision and causing the database to become inconsistent.

Such inconsistencies can be even less directly detectable. Consider the slightly modified partial program *Invest* with database Δ_2 :

```
(<= (Good Bonds Invest $h)
     (Holds (Standing Gorbachev) $h))

(<= (Good Stocks Invest $h)
     (Holds (Sitting Gorbachev) $h))

(<= (Holds (Sitting Gorbachev) $h)
     (Holds R (Next $h)))
```

This program specifies that the agent should invest in bonds if Gorbachev is standing up now, otherwise it should invest in stocks if Gorbachev is sitting down. The proposition *R* is not perceivable. Let's say the agent chooses to invest in *Bonds* at time t_4 . In some cases, the proposition *R* may turn out to be entailed by the agent's database at time t_5 (due to effects of actions, new perceptual information, or being informed by other agents), which in turn implies that the appropriate action at time t_4 should have been buying stocks.

Problems can also arise had we allowed arbitrary conditions to be used in specifying actions. Let *B3* be a partial program with the following database:

```
(<= (Good A1 B3 $h) FLT)
(<= (Good A2 B3 $h) (NOT FLT))
```

The predicate *FLT* denotes that the Fermat's Last Theorem is true. In theory, this program is uniquely defined despite the fact that nobody knows to date which is the intended behavior function. As a result, no interpreter can guarantee the correct execution of such a

program. We avoid this type of problems by requiring action specifications to be conditioned on information that is either in the partial program database or can be obtained through perception.

Another problem with the interpreter is that it is not complete. There are partial programs with non-empty extensions on which the interpreter will loop indefinitely without executing any actions. Due to the semi-decidability of first order logic, if $\beta(\eta) \neq \gamma$ is not logically implied by the database, any of the attempts to prove an action being inconsistent can run forever. In order to find a finite proof or a finite failure within a finite time, the inferences for proving the prescription or prohibition of each action must be interleaved.

3.3 INCREMENTAL PROGRAM DEVELOPMENT

Partial programs facilitate incremental program development using incomplete information. A programmer can start with a minimal specification about an agent and its behavior and gradually add more constraints. Let's continue with our partial program for the Square World as described in Section 2.2. If the database is augmented with the following:

```
(<= (Good $a Square-BF $h)
     (Holds (Next-to R32 G17 $a) $h))

(<= (Better Move-Clockwise
      Move-Counterclockwise
      Square-BF $h)
     (NOT (Provable (Bad Move-Clockwise
                      Square-BF $h))))
```

Given such a program, the agent would normally move clockwise as the program in Figure 2. However, whenever the agent is told that the gold is next to it via a counterclockwise movement, it will choose that action and thus executing the sequence of actions shown in Figure 3. We can also add to the database a new constraint so that the agent will avoid choosing an action whenever there is an obstacle blocking its execution, and thus resulting in the behavior shown in Figure 4.

```
(<= (Bad $a Square-BF $h)
     (Holds (Obstacle $a) $h))
```

4 PROPERTIES OF PARTIAL PROGRAMS

In this section, we examine some of the special properties about partial programs.

4.1 MONOTONICITY

As was demonstrated in the example above, partial programming is a process of incrementally constrain-

ing the potential actions of an agent by adding information to its database. The behavior of an agent becomes increasingly constrained as information accumulates. More precisely, the extension of a partial program decreases *monotonically* as (consistent) new information is added to its database.

Theorem 2 *Given two partial programs $(\beta_1, \Pi, \Gamma, \Delta)$ and $(\beta_2, \Pi, \Gamma, \Delta \cup \nu)$ for the same agent, if $\Delta \not\models \nu$ and $\Delta \cup \nu$ is satisfiable, then*

$$\epsilon(\beta_2) \subseteq \epsilon(\beta_1).$$

Proof: Given that the new information ν is consistent with the original database, more conclusions can be drawn from $\Delta \cup \nu$ while all the old conclusions should remain true⁸. As a result, if $\Delta \models \beta(\eta) = \gamma$ then $(\Delta \cup \nu) \models \beta(\eta) = \gamma$. Similarly, if $\Delta \models \beta(\eta) \neq \gamma$ then $(\Delta \cup \nu) \models \beta(\eta) \neq \gamma$. By definition, if a behavior function $f \in \epsilon(\beta_2)$ there exists an interpretation \mathcal{I} such that $\mathcal{I}(\beta_2) = \mathcal{I}(\beta) = f$ and $\models_{\mathcal{I}} (\Delta \cup \nu)$. Since the programs β_1 and β_2 are defined over the same percepts and actions, we have $\models_{\mathcal{I}} (\Delta \cup \nu)$ implies $\models_{\mathcal{I}} \Delta$. It follows that there exists an interpretation \mathcal{I} such that $\mathcal{I}(\beta_1) = f$ and $\models_{\mathcal{I}} \Delta$. We conclude that f must be in the extension of β_1 . Therefore, $\epsilon(\beta_2) \subseteq \epsilon(\beta_1)$. \square

The equality in $\epsilon(\beta_2) \subseteq \epsilon(\beta_1)$ is necessary since it is not the case that every new piece of information will affect the behavior of an agent. In order to distinguish information that results in behavioral changes from the others, we sometimes call such information *procedural hints*. We define this characterization based on the notion of minimal support [Reiter and de Kleer, 1987]. In short, a piece of information is a procedural hint if it changes the minimal support of some action prescriptions for the agent.

We can also take bigger steps by combining two partial programs. The extension of a combined partial program is a subset of the intersection of the extensions of its constituent programs.

Theorem 3 *Given two partial programs $(\beta_1, \Pi, \Gamma, \Delta_1)$ and $(\beta_2, \Pi, \Gamma, \Delta_2)$ for the same agent. Let β_3 be the combined program $(\beta_3, \Pi, \Gamma, \Delta_1 \cup \Delta_2)$. If Δ_1 is consistent with Δ_2 , we have*

$$\epsilon(\beta_3) \subseteq \epsilon(\beta_1) \cap \epsilon(\beta_2).$$

Proof: If any interpretation \mathcal{I} satisfies all of the sentences in $\Delta_1 \cup \Delta_2$, then \mathcal{I} satisfies all of the sentences in Δ_1 and all of the sentences in Δ_2 . Hence, any behavior function that is in the extension of β_3 is in the extensions of both β_1 and β_2 . \square

In some cases, the containment is proper.

⁸The names of the behavior functions for the two programs are assumed to be the same, and can be referred to as simply β .

Theorem 4 There are partial programs $\langle \beta_1, \Pi, \Gamma, \Delta_1 \rangle$ and $\langle \beta_2, \Pi, \Gamma, \Delta_2 \rangle$ for the same agent such that the extension of the combined program $\langle \beta_3, \Pi, \Gamma, \Delta_1 \cup \Delta_2 \rangle$ satisfies

$$\epsilon(\beta_3) \subset \epsilon(\beta_1) \cap \epsilon(\beta_2).$$

Proof: Consider the program B1 with database $\{R, R \wedge Q \Leftrightarrow B1(H)=A\}$ and the program B2 with database $\{Q, R \wedge Q \Leftrightarrow B2(H)=A\}$. The program B1 has an interpretation in which Q is false; and, therefore, any procedure consistent with $B1(H) \neq A$ is in its extension. The program B2 has an interpretation in which R is false; and, therefore, any procedure consistent with $B2(H) \neq A$ is in its extension as well. Consequently, any such procedure is in the intersection of the two extensions. However, $B3(H)=A$ is true in every interpretation of the joint program, where B3 is the name of the combined program. Therefore, its extension does not contain any procedure consistent with $B3(H) \neq A$. \square

The significance of these results is that one must be careful in combining partial programs so that the assumptions in writing one program and the assumptions in writing the other program don't combine to eliminate procedures intended to be in the extension of the joint program. Interactions of this sort can be caught so long as the assumptions are explicitly recorded. However, if the assumptions are left implicit as the examples in the proof, the contradiction can go unnoticed and some intended procedures can be inadvertently lost.

4.2 COMPLETENESS

A program can be incomplete in the sense that, for some perceptual histories, actions are not uniquely prescribed or are undefined. For example, if $B1(H_1) \neq A_1$ (i.e. the action A_1 is prohibited for the partial program B1 at perceptual history H1) is the only information about B1, then any other action of the agent will be equally applicable at H1. Therefore, B1 is not complete. Intuitively speaking, a program is *complete* if there is exactly one action that is applicable at every perceptual history, or equivalently,

Definition 5 A program $\langle \beta, \Pi, \Gamma, \Delta \rangle$ is *complete* if

- It has exactly one behavior function in its extension, i.e. $|\epsilon(\beta)| = 1$.
- Every model of Δ that agrees with the given naming function should agree on the terms $\beta(\eta) = \gamma$, for all $\eta \in \Pi^*$ and $\gamma \in \Gamma$.
- For any two interpretations I and J that satisfy Δ , there exists a homomorphism $h : I \leftrightarrow J$, such that for all history η ,

$$(h(\beta^I(\eta^I)) = \beta^J(h(\eta^J))).$$

- For all history $\eta \in \Pi^*$ and action $\gamma \in \Gamma$, we have $(\Delta \models \beta(\eta) = \gamma) \vee (\Delta \models \beta(\eta) \neq \gamma)$.

4.3 DIRECTNESS

Sometimes, even though a unique action is prescribed for a given perceptual history by the partial program, the derivation of the prescribed action still requires more computation than simply looking up that fact in the database. For example, the two statements shown below entail a unique action A_2 for perceptual history H1, even though the action prescription is not directly specified by the program.

$$\begin{aligned} B6(H_1) &= A_1 \vee B6(H_1) = A_2 \\ B6(H_1) &\neq A_1 \end{aligned}$$

In this case, it is relatively easy to deduce the prescribed action. Suppose the chain of reasoning is long or branchy, it can be very expensive to compute the prescribed action. In general, the computation grows exponentially with the number of statements in the database Δ .

The notion of directness is defined with respect to an interpreter, since executing a partial program on different interpreters can result in very different efficiency (or even different behavior).

Definition 6 A partial program is *direct* with respect to an interpreter if the choice of action at every time point can be computed within a bounded time.

As the interpretation of partial programs involves theorem proving, efficiency issues are of great concerns in making partial programs a practical tool. It is desirable to compile a partial program into increasingly more direct forms so that its runtime performance can be improved. Because of the expressiveness of the language, there exist partial programs that do not have any compatible direct programs. In [Hsu, 1991], we describe an algorithm for synthesizing efficient execution programs from given partial programs for a restricted class of partial programs. Other compilation techniques have been developed to improve the performance of a partial programming system and can be found in [Hsu, 1991].

4.4 PARTIAL vs. NONDETERMINISTIC PROGRAMS

At a first glance, nondeterministic programs seem to resemble partial programs since they both describe sets of deterministic procedures. However, there is a subtle but significant difference between the two. Nondeterministic programs explicitly express the intended ambiguity in action selection (i.e. a stated lack of preferences), while partial programs implicitly assume no preferences among actions unless stated otherwise (i.e.

a lack of stated preferences). Much research in computer science has been concerned with the invention of control primitives for expressing the former, which is sometimes called don't-care nondeterminism. The latter is more similar to the so-called don't-know (or incidental) nondeterminism except that it is also dynamic in nature. As a result of the constant accumulation of information, some of the choices may become fixed as the agent functions in its environment. This distinction gives rise to a very important difference in their expressive power, which is elaborated by the following two theorems.

Theorem 5 *For every complete nondeterministic program, there is a partial (deterministic) program with the same extension.*

Proof: Given a complete nondeterministic program $\langle \rho, \Pi, \Gamma, \Delta \rangle$, where ρ is a relation symbol denoting the behavior relation for the agent. The set of actions that may be executed at history η is represented by $\rho(\eta)$. We can construct a corresponding partial program $\langle \beta, \Pi, \Gamma, \Delta' \rangle$ by selecting a new function constant β and augmenting the database as follows:

$$\Delta' = \Delta \cup \{\forall \eta \forall \gamma \gamma \notin \rho(\eta) \Rightarrow \beta(\eta) \neq \gamma\}.$$

□

Theorem 6 *There is a partial (deterministic) program for which there is no complete nondeterministic program with the same extension.*

Proof: This can be shown by an example. Given an agent with exactly two actions $\Gamma = \{A1, A2\}$. Let's suppose that its behavior is specified by the partial program $\langle B, \Pi, \{A1, A2\}, \{B(H1) = B(H2)\} \rangle$, where $H1$ is a prefix of $H2$, i.e. $H2$ is a history consists of the history $H1$ followed by a non-zero sequence of percepts. For any interpretation that maps B into some behavior function $f \in \epsilon(B)$, either $B(H1) = A1$ and $B(H2) = A1$ are both true or $B(H1) = A2$ and $B(H2) = A2$ are both true. In order for the extension of a complete nondeterministic program $\langle B', \Pi, \{A1, A2\}, \Delta' \rangle$ to admit these interpretations, it must be the case that $\Delta' \models (B'(H1) = \{A1, A2\})$ and $\Delta' \models (B'(H2) = \{A1, A2\})$. But then there exists a behavior function f' in the extension $\epsilon(B')$ such that $f'(h_1) = a_1$ and $f'(h_2) = a_2$. □

5 PREVIOUS WORK

The idea of programming a machine, whose problem solving behavior results from manipulating sentences in a formal language, so that it is possible to change the machine's behavior by giving declarative instructions to it was first proposed by McCarthy in his "Advice Taker" paper [McCarthy, 1968]. Later, Hewitt's Planner [Hewitt, 1972] provided some limited capability of giving imperatives to a system, and Hayes further advocated the view of computation as controlled

deduction [Hayes, 1973]. Unfortunately, despite their influences on the field of AI, none of the systems were fully explored.

Similar ideas for separating a program into its logic (what) and control (how) were a central theme in logic programming [Kowalski, 1979a]. As a declarative language, Prolog provides a straightforward way to express the *what* part of a problem [Clocksin and Mellish, 1984; Kowalski, 1979b]. However, there is little support for representing control explicitly⁹. Procedural information is often implicitly encoded as cuts and ordering of clauses and conjuncts in a Prolog program, which makes modifying a program much more difficult.

Partial programs, on the other hand, provide a way for specifying both what the problem is and how the problem should be solved in a unified framework. As a result, a machine is able to take advantage of new information in order to achieve its goals more efficiently and effectively.

We should also point out that although the syntactic forms of partial programs do not look very different from a production system (or other rule based systems), there are several significant differences: 1) Historical information are taken into account, so partial programs are more expressive. 2) The semantics of partial programs enables one to program by subtraction, while a production system does not have such nice monotonic properties. 3) In a production system, there is no way to say certain actions should *not* be performed under certain conditions.

6 CONCLUDING REMARKS

We have presented partial programs as a new framework for incremental program development in complex domains. A partial programming language allows one to write programs without making premature commitments, and its flexibility facilitates incremental integration of control information in designing efficient programs. The practicality of partial programs have been tested in domains such as automated diagnosis for computer hardware faults [Genesereth, 1985], simulation of digital hardware [Singh, 1983] and some others. More recently, we've applied partial programming to the domain of planning and constraint satisfaction problems. In [Hsu, 1990], we showed that partial programming can be used as a least-commitment strategy, and together with an anytime algorithm [Dean and Boddy, 1988] an agent is able to exhibit reactive behavior [Georgeff and Lansky, 1987, Agre and Chapman, 1987]. We plan to conduct more experiments in applying partial programming to bigger-scale real world domains.

⁹Work on meta-interpreters has attempted to address this issue to some extent.

The most important issue for current and future research in partial programming is in the area of performance improvement. The compilation of partial programs is complicated by two factors: First, the mapping between a partial program and the compiled program is not unique. Second, the dynamic nature of information in a partial programming system. The work in [Hsu, 1991] provides solutions to some points in the entire space of compilation problems, and further exploration is expected make partial programs a even more powerful tool.

Acknowledgements

The authors would like to thank Narinder Singh, John Woodfill, Devika Subramanian, and Nils Nilsson for helpful discussions. Support for this research was provided by the Office of Naval Research under contract number N00014-81-K-0004 and grant number N00014-90-J-1533.

References

- [Agre and Chapman, 1987] P. E. Agre and D. Chapman. PENG: An implementation of a theory of activity. In: *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA (AAAI 1987) 268-272.
- [Clocksin and Mellish, 1984] W. F. Clocksin and C. S. Mellish. *Programming in Prolog* (Springer-Verlag 1984).
- [Dean and Boddy, 1988] T. Dean and M. Boddy. An analysis of time-dependent planning. In: *Proceedings of the Seventh National Conference on Artificial Intelligence*, St. Paul, MN (AAAI 1988) 49-54.
- [Epikit, 1989] *The Epikit Manual*. Epistemics, Inc. 1989.
- [Fikes and Nilsson, 1971] R. F. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2 (1971) 189-208.
- [Genesereth et al., 1990] M. R. Genesereth et al. Knowledge interchange format: Version 2.0 Reference manual. Technical Report LOGIC-90-4, Computer Science Department, Stanford University (1990).
- [Genesereth, 1985] M. R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence* 24 (1985) 411-436.
- [Georgeff and Lansky, 1987] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In: *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA (AAAI 1987) 677-682.
- [Hayes, 1973] P. J. Hayes. Computation and deduction. In: *Proceedings of the Symposium on Mathematical Foundations of Computer Science* (Czechoslovakian Academy of Sciences 1973) 105-117.
- [Hewitt, 1972] C. F. Hewitt. Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot. Technical Report TR-258, MIT AI Laboratory (1972).
- [Hsu, 1990] J. Y. Hsu. Partial planning with incomplete information. In: *Proceedings of the Pacific Rim International Conference on Artificial Intelligence '90*, Nagoya, Japan (1990) 474-479.
- [Hsu, 1991] J. Y. Hsu. *Partial Programming - A Framework for Incremental Problem Solving*. PhD Dissertation (forthcoming), Computer Science Department, Stanford University.
- [Kowalski, 1979a] R. Kowalski. Algorithm = logic + control. *Communications of the ACM* 22 (1979) 424-436.
- [Kowalski, 1979b] R. Kowalski. *Logic for Problem Solving* (North Holland, New York, NY 1979).
- [McCarthy, 1968] J. McCarthy. Programs with common sense. In: M. Minsky, editor, *Semantic Information Processing* (MIT Press, Cambridge, MA 1968) 403-418.
- [Reiter and de Kleer, 1987] R. Reiter and J. de Kleer. Foundations of assumption-based truth maintenance systems: preliminary report. In: *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle WA (AAAI 1987) 183-188.
- [Shoham and Goyal, 1988] Y. Shoham and N. Goyal. Temporal reasoning in artificial intelligence. In: *Exploring Artificial Intelligence*, Shrobe, H. E. and AAAI eds. (Morgan Kaufmann, CA, 1988) 419-438.
- [Sikka et al., 1991] V. Sikka, M. R. Genesereth, and N. P. Singh. Paralog: A Fast Algorithm for Automatic Theorem-Proving with Equality. Technical Report LOGIC-91-2, Computer Science Department, Stanford University (1991).
- [Singh, 1983] N. P. Singh. MARS: A multiple abstraction rule-based simulator. Knowledge Systems Laboratory Report KSL-83-43, Stanford University (1983).

Computational Considerations in Reasoning about Action

Matthew L. Ginsberg
 Computer Science Department
 Stanford University
 Stanford, CA 94305

Abstract

Any first-principles declarative planner will need to be constructed from an underlying declarative system that reasons about action. In this paper, we point out that if such a planning system is to be computationally viable, the associated declarative description of actions must satisfy at least two broad conditions. First, it will need to be *event-driven*, so that fluents that hold in a particular situation can be propagated into the future at reasonable computational cost. Second, it will need to be *anytime* in the sense that partial or approximate answers to queries can be provided in the presence of computational resource constraints. We suggest that the first of these goals can be achieved by taking the truth values assigned to sentences to be functions from the temporal elements into a set of basic values, and that the second can be achieved by viewing temporal operators as functions on these functional truth values.

1 INTRODUCTION

Existing planning systems can be grouped into three broad categories: expert planners, general-purpose planners, and first-principles planners.

Expert planners, of which there are many, are essentially applications of expert-systems technology to planning problems. The situation in which a particular agent finds itself is classified to determine which of a predetermined set of actions is most likely to be effective in achieving the agent's goals. There has been some interest recently in constructing the expert decision rules automatically [Drummond, 1988], but the approach itself must inevitably be limited by the fact that the agent involved has no real idea what's going on – it is simply mindlessly applying rules that govern its behavior. The ensuing brittleness is typical of expert systems generally.

General-purpose planners, of which there are few, attempt to address this difficulty by working with a set of action descriptors that describe the possible actions in some domain, and then constructing a plan to achieve a particular goal using methods that are independent of the domain in which the agent finds itself. This work began with STRIPS [Fikes and Nilsson, 1971]; the most successful existing planner of this sort is arguably the SYPE system [Wilkins, 1988].

There are two difficulties with the general-purpose approach. The first is that the computational complexity of planning problems is typically very high, making it impractical to generate a complete plan that is guaranteed to achieve a particular goal. SYPE addresses this difficulty by restricting the form of the actions it can consider.

The nondeclarative restrictions placed on the form of the actions being considered unfortunately make these planners nonuniversal; there are domains for which any particular restriction is inappropriate. This is the essence of the second difficulty: General-purpose planners, by committing at a fundamental level to a specific description of actions, inherit some (but by no means all) of the brittleness of their expert-planning predecessors. "General-purpose" planners are only general-purpose within the bounds established by assumptions embodied in the form of the action descriptors.

First-principles planners (of which there are none) attempt to address these difficulties by viewing planning as a purely declarative activity. More specifically, first-principles planners view planning as theorem proving set against the background of a declarative system that describes actions in a particular domain.

This idea is an old one, dating back to the QA3 system [Green, 1969]; as work on declarative systems generally has advanced, the attractiveness of the approach has remained. With the development of nonmonotonic reasoning, for example, it was suggested that this general declarative notion could be applied to construct a planner that would be able to jump to conclusions while building its plans. It was later suggested that

assumption-based truth maintenance [de Kleer, 1986], another general declarative technique, might bear on the problem of debugging plans that appear to be non-monotonically sound but that closer inspection reveals to be flawed in some way [Ginsberg, 1990b].

The reason that there are no established planners of this type is that the underlying declarative descriptions of action are themselves lacking. The best-known reason for this is the infamous Yale shooting problem [Hanks and McDermott, 1987], although a variety of researchers have found solutions to this particular difficulty.

A more fundamental problem with declarative descriptions of action is that they are simply unsuitable for inclusion in planners. The approach suggested in [Green, 1969] and reiterated in [Genesereth and Nilsson, 1987] is still a valid one – given a monotonic description of a domain, it is indeed possible to view planning as theorem proving. The difficulty is that it is not practical to do so.

The reason for this can be seen by considering the frame axiom. Here is a typical rendering of it that might be included in a nonmonotonic axiomatization:

$$\text{holds}(f, s) \wedge \neg \text{ab}(a, f, s) \supset \text{holds}(f, \text{result}(a, s)) \quad (1)$$

Informally, this axiom says that if some fluent f holds in a situation s and the action a is not abnormal in that it reverses f when executed in the situation s , then f will continue to hold after the action is completed.

There are technical problems with this definition, but they can be avoided [Baker and Ginsberg, 1989, and many others]. But an overwhelming computational difficulty can be seen if we imagine using (1) to propagate a set of fluents through a long sequence of actions. The application of (1) for each action and to each fluent will result in a prohibitively large number of consistency checks, making the system unusable in practice.

This problem is avoided in general-purpose planning systems by using a nondeclarative description of action that has more attractive computational properties. In STRIPS, for example, actions are described in terms of add and delete lists, reducing the complexity of the reasoning enormously. The STRIPS formalism cannot deal with the inferred consequences of actions, however, as was observed in [Lifschitz, 1986]. (This is called the *ramification problem* in [Finger, 1987].) A partial solution to this difficulty can be found in [Ginsberg and Smith, 1988], but the approach presented there continues to describe actions in nondeclarative terms.

The intellectual foundation for the work described in this paper lies in an attempt to present a declarative description of the work in [Ginsberg and Smith, 1988]; we have tried to develop a formalization of action that

will be computationally viable in the situations likely to arise in planning. The two specific heuristic commitments that we will make are the following:

First, we will assume that fluents typically survive long sequences of actions before being needed; a robot should be able to put a wrench in its toolbox, perform most of its day's activities, and conclude at a single stroke that the wrench is still in the toolbox. We will describe this by saying that our formalization of action needs to be *event-driven* in the sense that propagating fluent values through idle periods not incur significant computational expense.

Second, we will commit ourselves to a system that can reason about actions in an *anytime* fashion; the word appears to originate in [Dean and Boddy, 1988]. By this we mean that the system, when asked the value of a fluent in a specific situation, will produce some answer quickly, perhaps modifying that answer as necessary if allowed to consider more subtle features of the situation involved. It is generally recognized that planning problems are sufficiently difficult that approximate answers are inevitable; we are simply requiring that this sort of computational response be present in the description of action that underlies the planner itself.

The reason that we have chosen to discuss these two problems in this paper is not that there are no others (there are), but that the solutions to them are linked. Roughly speaking, both difficulties can be addressed by taking the truth value assigned to a sentence to be not a single value such as "true" or "false," but a function from a set of time points into such values.

The reason that this approach leads to an event-driven description is that it allows us to conveniently describe the expected future behavior of fluents in a compact fashion. Instead of saying, "The wrench is in the toolbox at 9:15," and, "Things in toolboxes tend to remain there," we can simply say, "The wrench is expected to be in the toolbox for the rest of the day," meaning that the truth value assigned to the sentence

$$\text{in}(\text{wrench}, \text{toolbox})$$

is a function that maps the entire temporal interval from 9:15 to 5:00 to the value t (or perhaps dt – true by default – if we are prepared to admit the possibility of subsequent information reversing our conclusions). The problem of making our description event-driven now becomes essentially a matter of finding a data structure for the functional truth values that efficiently encodes the behavior of fluents that change only infrequently.

The idea of taking truth values to be temporal functions also bears on our requirement that the implementation of our formalism exhibit anytime behavior. As an example, consider a sentence such as, "One second after the valve is closed, the pressure will increase,"

which we will write somewhat schematically as

$$\text{delay(closed-valve,1)} \supset \text{pressure} \quad (2)$$

where `delay` is an operator that we will use to push the temporal description of the valve into the future. The second argument to `delay` in (2) is a parameter indicating the amount by which the truth value is to be displaced.

From a formal point of view, the `delay` operator appearing in (2) is a *modal operator*, since one of its arguments is not an object in our language but a declarative sentence instead. It is shown in [Ginsberg, 1990a] that when truth values are taken to be more descriptive than simply elements of the two-point set $\{t, f\}$ (true and false, respectively), it is possible to view modal operators as functions on the truth values of their propositional arguments. In (2), the modal operator `delay` corresponds to the *function* `delay` that is given by

$$[\text{delay}(f, \Delta)](t + \Delta) = f(t) \quad (3)$$

Note that `delay` accepts a function as its first argument and returns a function as its result, since the truth values that we are using are themselves functional.

To see that this interpretation leads to anytime behavior, we need to make one more observation: The basic purpose of a deductive system is to determine what truth value should be assigned to a particular query. Now note that when considering a query q , we may well encounter a modal operator, requiring us to apply the corresponding function (as in (2) or (3)) to the truth values of the propositional arguments (`closed-valve` in (2)). But what are we to use for these truth values? We can use either the result of invoking the theorem prover recursively on the propositional arguments themselves, or use the values that can be obtained by simply searching for the given propositions in the database. Using these latter values as approximations to the former leads to a system that produces some answer quickly, but may modify that answer on further consideration. Perhaps there is a deductive demonstration that the valve in (2) will be open at some particular future time, and so on. If the analysis of the embedded sentences produces still further modal expressions, anytime behavior will result as the system makes and then examines assumptions about the truth values assigned to these embedded sentences. This observation is expanded upon in [Ginsberg, 1991a].

The remainder of this paper will consider each of these ideas in turn, and then show an example of an implementation of them being used to analyze a shooting scenario similar to that appearing in [Hanks and McDermott, 1987]. The implementation is built using the multivalued theorem proving system `MVL` [Ginsberg, 1988, Ginsberg, 1989, Ginsberg, 1991b].

2 TRUTH VALUES

We remarked in the introduction that we intend to label sentences in our declarative database with functional truth values that include information about the truth or falsity over time (or default truth/falsity, etc.) of the sentence involved. The reason that we are comfortable doing this is that the labels so constructed retain the mathematical structure of the original “instantaneous” labels, in that we can combine them, negate them, disjoin and conjoin them, and so on.

It is these operations of conjunction, disjunction and so forth that underlie the semantics of any declarative system. Specifically, if we have labels x and y for sentences p and q respectively, we need a way to construct a label $x \vee y$ for the disjunction of p and q , and so on. In [Ginsberg, 1988], a mathematical structure called a *bilattice* was introduced that consists of a set together with just these combining operations. Although the formal definition will not play a significant role in the remainder of this paper, we include it here in the interests of completeness:

Definition 2.1 A bilattice is a sextuple $(B, \wedge, \vee, \cdot, +, \neg)$ such that:

1. (B, \wedge, \vee) and $(B, \cdot, +)$ are both complete lattices, and
2. $\neg : B \rightarrow B$ is a mapping with $\neg^2 = 1$ such that \neg is a lattice homomorphism from (B, \wedge, \vee) to (B, \vee, \wedge) and from $(B, \cdot, +)$ to itself.

The bilattice operations \wedge , \vee and \neg all correspond to the usual logical notions, while $+$ corresponds to combination of evidence and is used to combine the truth values obtained from separate lines of reasoning to a single conclusion. Many more details of the bilattice work, together with a discussion of the philosophical ideas underlying the approach, can be found in [Ginsberg, 1988].

What is of interest to us here is the fact that if B is some bilattice, then B^2 , the collection of ordered pairs of elements of B , inherits a bilattice structure from B where all of the bilattice operations are computed pointwise. (The construction is analogous to the construction of the Cartesian plane \mathbb{R}^2 as the product of two copies of the real line.) More generally, for any set S , the set B^S of functions from S into B inherits a bilattice structure from the set B .¹

It follows that if we have some set T of time points, then the set B^T of functions from T into the “base” set of truth values B has the structure required of a set

¹There is no real difference between viewing the set B^2 as the set of ordered pairs of elements of B , or as the set of functions from the two-point set $\{1, 2\}$ into B .

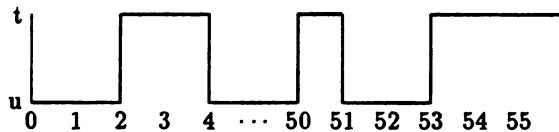


Figure 1: A fluent that is true three times

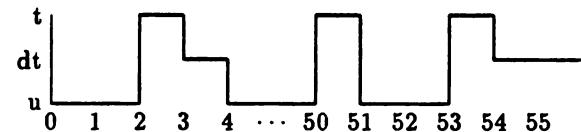


Figure 2: A default frame axiom

of truth values. As an example, if we take T to be the integers, then the graph in Figure 1 shows the truth value assigned to a fluent that is true for two units of time at $t = 2$, for one unit of time at $t = 50$, and for all time after $t = 53$.

Our event-based philosophy now corresponds simply to a data structure that represents these truth values by listing the points at which the value changes. In Figure 1, for example, we record the fact that the fluent is unknown at time 0, true at time 2, unknown at time 4, and so on; values at a total of six points are recorded. Determining the value of the fluent at any intermediate time t is a matter of walking along the graph until the next event is later than t , and taking the value at the last time encountered. Note that the computational effort required to determine the value of the fluent in Figure 1 is completely independent of the length of the gap between times 4 and 50.

2.1 EXTENSIONS

The ontological shift that we are proposing does not in and of itself commit us to any specific computational or representational strategy. As an example, the simple representation scheme that we described in the previous paragraph can easily be extended in a variety of ways:

1. The set T of time points only needs the structure of a partial order in order for the above approach to work; to determine the value of a fluent f at some particular time t , we walk our way along the function until we find ourselves between two points t_0 and t_1 such that

$$t_0 \leq t < t_1,$$

so that t is no earlier than t_0 and t_1 is later than t . The value of f at t is then the value taken at t_0 .

2. As an example, taking the above partial order to be the continuous real line allows us to avoid our earlier implicit assumption that time was discrete. This particular choice commits us to fluents being true over half-open intervals $[x, y)$ only, but this can be avoided by introducing auxiliary elements x^+ for each $x \in \mathbb{R}$ such that the half-open interval $[x, y^+)$ in fact denotes the closed interval $[x, y]$.
3. Another example involves taking the elements of the partial order to be action sequences, where an action sequence a_2 temporally follows a sequence

a_1 whenever a_2 is an extension of a_1 . Nonlinear action sequences can be handled by weakening the partial order to cater to possible linear action sequences consistent with a given nonlinear one. Alternatively, by using sequence variables to denote arbitrary sequences of actions following some particular one, it is possible to use the partial order given by variable instantiation to capture this temporal information. This idea appears to be central to the use of our ideas in a functioning planning system.

4. It is also possible to extend the scheme by introducing “decay functions” that describe how a fluent’s truth value is expected to change as time goes by. In Figure 1, for example, the fluent’s truth values do not change at all as time passes; a more realistic example might involve the truth value of the fluent falling from t to dt at times 3 and 54, as shown in Figure 2. Here, our confidence in the truth of the fluent decays as time passes, corresponding to the application of a non-monotonic frame axiom. As before, information is recorded only when the truth value of the fluent changes from the expected one, so we still need to record information only about the “events” at times 0, 2, 4, 50, 51 and 53. By changing the set of base truth values to which the temporal functions map, this idea can be extended to include a wide variety of temporal behaviors, such as the probabilistic decay functions discussed in [Dean and Kanazawa, 1988]. A fuller discussion of this point can be found in [Ginsberg, 1991c].

The computational efficacy of the scheme that we have proposed is preserved in all of these cases.

2.2 THE FRAME AXIOM

The approach we have suggested encodes the information that is normally captured by a frame axiom such as (1) in two separate ways. First, the actual default behavior assigned to some particular fluent f is encoded in a truth value such as that appearing in Figure 2, which explicitly indicates the default truth of the fluent at times following times when it is known to be true with certainty.

We have thus far avoided the question of how we obtain truth values such as that appearing in Figure 2 in the first place. What we would like to do is to have a sentence in our database such as the following:

If the robot puts a tool in its toolbox, then the tool is definitely in the toolbox at that time, and can be assumed by default to be in the toolbox at subsequent times.

Note first that the truth value to be assigned to the conclusion of the rule (that the tool is in the toolbox) is not the same as the truth value of the premise; the premise (that the tool is being put into the toolbox) holds only instantaneously, while the conclusion (that the tool is actually located in the toolbox) holds over a wide range of times. This is a technical issue that we will consider in the next section and is identical in principle to the appearance of the delay operator in the introduction.

More importantly, note that the future behavior of any particular fluent (in this case, the location of the tool) is determined not by applying some blanket frame axiom such as (1), but instead by an axiom describing this future behavior when the fluent is first asserted (i.e., when the robot puts the tool in the toolbox).

This is an important distinction between our approach and the conventional one. Computational issues aside, we prefer our approach on purely philosophical grounds, since it is *not* the case that the frame axiom applies to all, or even most of the fluents we encounter in everyday life.

As an example, consider the problem of entering a crosswalk when there is a car five feet away approaching at 60 MPH. Are we to apply the frame axiom to the fact that the car is five feet from the crosswalk, or to the fact that it is moving at a high speed? Clearly to the latter, although there is no information in (1) indicating that we should do so.

This is related to the well-known *problem of induction* [Skyrms, 1966]. How is it that we know to apply the frame axiom to a predicate such as "blue" or "green" but not to one such as "grue" (green until July 10th but blue subsequently) or "bleen?" Although we have not provided an answer to this question, we have indicated clearly the declarative point at which such an answer is used – in the description of the expected future behavior of newly established fluents. Similar observations have also been made in [Myers and Smith, 1988].²

²[Haas, 1987] and [Schubert, 1990] have also argued against blanket frame axioms, suggesting instead that one should reason from specific fluent changes to the occurrence of an action of some sort. It seems to me, however, that this approach is unable to cope with either the ramification problem or with the moving automobile described earlier in this section.

3 MODAL EXPRESSIONS

Let us return to the observation made in the last section that the truth value to be assigned to the consequent of some rule is often not the same as the truth value of the antecedent. In the introduction, we handled a situation such as this in (2) by introducing a modal operator *m* and writing

$$m(a) \supset c$$

where *a* is the antecedent and *c* is the consequent. The modal operator *m* changes the truth value of *a* so that the truth value of *c* is modified correctly by the above rule. The example in the introduction involved a modal operator *delay* that delays the truth value of the antecedent by some amount of time; our current concerns can be dealt with by introducing an operator *propagate* that is responsible for inserting the consequent into the database with a complete "future history" if appropriate.

We will discuss these two operators in some detail shortly, but let us continue to examine general issues first. The idea that modal operators can be viewed truth-functionally (i.e., as functions on the truth values of the sentences on which they operate) is an old one in the philosophical community, but was discarded in favor of the possible-worlds approach [Kripke, 1971] when it was realized that there simply are not enough functions on the two-point set {t, f} to correspond to all of the interesting modal expressions that one might wish to consider.

In [Ginsberg, 1990a], however, it was pointed out that if truth values are taken from an arbitrary bilattice instead of from the set {t, f}, it becomes practical to view modal operators truth-functionally after all; in fact, the resulting construction is a generalization of Kripke's.

3.1 ANYTIME BEHAVIOR

An additional advantage of viewing modal operators truth-functionally is that the associated declarative systems naturally exhibit anytime properties; this can probably be made clearest by an example from PROLOG. Consider the following program:

```
landlubber(X) :- animal(X), not(fly(X)).  
fly(X) :- bird(X), not(penguin(X)).  
animal(X) :- bird(X).  
penguin(X) :- bird(X), tuxedo(X).  
bird(opus).  
tuxedo(opus).
```

Animals that cannot fly are landlubbers, birds can fly unless they are penguins, and birds in tuxedos are penguins. Opus is a bird wearing a tuxedo. Is he a landlubber?

Ignoring inadequacies in our representation of the domain, the interpreter begins by noting that it can prove that Opus is an animal, and therefore that he is a landlubber unless he can be shown to fly. A new proof process is therefore begun with the intention being to prove that Opus can fly.

Since Opus is a bird, he can fly unless he can be shown to be a penguin. Yet another proof process is begun; since this one succeeds in showing Opus to be a penguin, he cannot be shown to fly and the original query (is Opus a landlubber?) succeeds.

[Ginsberg, 1990a, Ginsberg, 1991a] propose that it be possible to interrupt this procedure at the points where new proof attempts are generated. Thus when creating the attempt to prove that Opus can fly, we note that since there is nothing in the database indicating explicitly that he can, we can tentatively label

`fly(opus)`

as unknown, and therefore assign `not(fly(opus))` the value of true using PROLOG's negation-as-failure rule. This allows us to tentatively confirm the original query.

Given more time, we can work on the goal `fly(opus)`, noting that this spawns the subgoal `penguin(opus)`. Once again, we break the inference process, using the fact that `penguin(opus)` is missing from our database to conclude tentatively that `not(penguin(opus))` is true and therefore that Opus can fly, so that the original query should fail. Finally, given still more time for reflection, we realize that Opus is a penguin and therefore a landlubber after all.

PROLOG's treatment of negation is as an operator that returns `t` unless the truth value of the argument is itself `t`; specifically, if some sentence `p` is unknown, negation-as-failure treats `not(p)` as true. Viewed in this fashion, PROLOG's negation is a modal operator in our sense. We are proposing two extensions to this idea:

1. Extending the notion of a modal operator to include temporal operators such as those that arise when reasoning about action.
2. Using these modal operators as semantic markers for points at which the inference process can be suspended and an approximate answer computed.

The first of these is the focus of the current paper; the second is treated implicitly in [Ginsberg, 1990a] and in more detail in [Ginsberg, 1991a].

3.2 TEMPORAL OPERATORS

Given that we take the view that temporal operators can be described by giving their functional behavior and then incorporating them into our declarative language, what operators are required in a system that reasons about action?

We will clearly need an operator `delay` to separate the occurrence of an action from the appearance of its effects, and another operator `propagate` that allows us to construct temporal truth functions such as the one appearing in Figure 2.

In fact, we appear to need nothing else; the axiomatisation appearing in the appendix uses no modal operators other than these two. The description of `delay` is as appearing in (3), while `propagate` is defined recursively as³

$$[\text{propagate}(f)](t) =$$

$$\begin{cases} f(t), & \text{if } t = 0 \text{ or} \\ & f(t) \neq u; \\ \text{decay}[\text{propagate}(f)(t - 1)], & \text{otherwise.} \end{cases}$$

The function `decay` might be given by, for example:

x	$\text{decay}(x)$
t	dt
dt	dt
f	df
df	df
u	u

This decay function maps any truth value into its default version, corresponding to a nonmonotonic frame axiom. A monotonic frame axiom would simply take $\text{decay}(x) = x$. As an example, Figure 3 shows the result of applying `propagate` to a temporal function that changes from t to f . Figure 4 shows the result if the frame operator is chosen to be monotonic.

From these two operators, we can build a declarative description of action that has the desired properties of being event-driven and anytime. If a is an action that causes a fluent f to be true in a persistent way (such as putting a tool in the toolbox), we write⁴

$$\text{duration}(a, t) \wedge \text{propagate}(\text{delay}(a, t)) \supset f \quad (4)$$

while if a causes f to be true only instantaneously (like striking a match causing a light), we write

$$\text{duration}(a, t) \wedge \text{delay}(a, t) \supset f$$

³The implementation of `propagate` does not follow this definition directly, since this would be horrendously inefficient. Instead, we use a monotonic propagation function (as described in the appendix), and simply drop information about any "events" at which a truth function becomes unknown.

⁴The axiom (4) is not quite satisfactory as it stands, because it is awkward to combine it with domain constraints describing ramifications of the action in question. This is handled in the appendix by reifying the fluents so that they can be treated as objects in our language, and replacing (4) with an axiom like

$$\begin{aligned} \text{causes-persistently}(a, f) \wedge \text{duration}(a, t) \wedge \\ \text{propagate}(\text{delay}(\text{holds}(a, t))) \supset \text{holds}(f) \end{aligned}$$

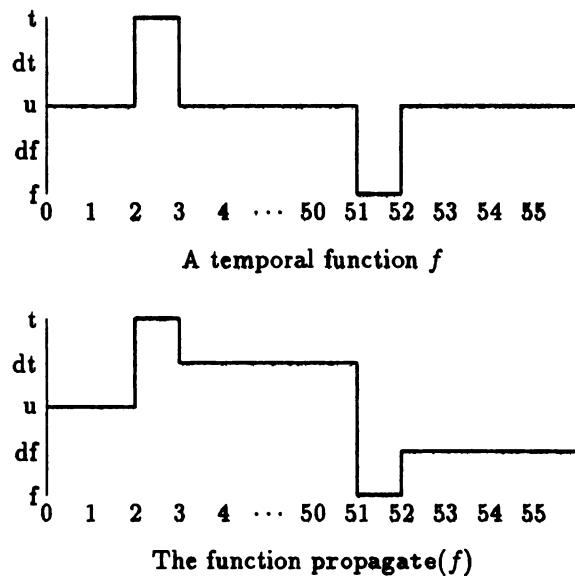


Figure 3: Applying the frame axiom

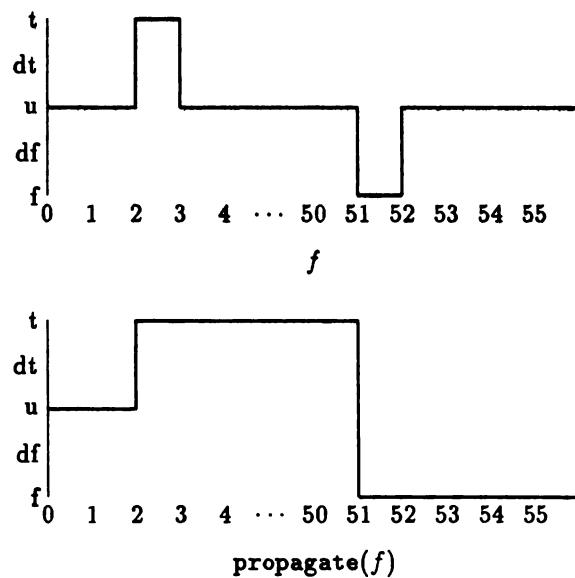


Figure 4: A monotonic frame axiom

The predicate **duration** indicates the amount of time taken by any particular action and is used to determine the value to be assigned to the parametric argument of **delay**.

3.3 EFFICIENCY CONSIDERATIONS

Suppose that we consider the example in Figures 3 and 4 a bit more closely. In the example in Figure 4, for example, note that there is no point to attempting to show that the fluent holds at time 4 – we already know this by virtue of the application of our monotonic propagation function. This implies the following:

When invoking the prover on a sentence p in order to evaluate a modal expression $m(p)$, one should only investigate proofs that will affect not only the truth value of p , but the truth value of $m(p)$ as well.

In fact, the situation is a bit more subtle still; consider Figure 3.

Suppose that we are interested in showing that the fluent f holds at time 53, perhaps because f is a precondition to an action that we would like to take at that time, or perhaps because the restriction mentioned above implies that this is the only information about f that is of interest to us.

In the initial situation in which we know nothing, it follows that we should try to show that f holds at any time before $t = 53$, since this value will then be propagated to the time of interest. Although showing that f holds at time 54 does effect the value of $\text{propagate}(f)$, it does not do so in an interesting way. This means that we should replace the above principle with the following stronger one:

When invoking the prover on a sentence p in order to evaluate a modal expression $m(p)$, one should only investigate proofs that will change the truth value of $m(p)$ in a way that will affect the response to the original query.

Applying this idea can be fairly subtle. In the example we are considering, suppose that we succeed in showing that f holds at time 2, so that $\text{propagate}(f)$ holds by default at time 53. Now there is no point in showing that f holds, but there is a reason to show that the negation of f holds at some time between 3 and 53, since $\neg f$ will block the propagation of f to the time that is of interest to us.

In the example in the figure, we can show that $\neg f$ holds at time 51; now we must once again change the focus of our proof efforts as we attempt to show that f is true either at $t = 52$ or at $t = 53$.

From a conceptual point of view, this is all quite straightforward. From an implementational point of

view, however, it can be rather subtle, especially since we should preserve portions of the proof tree for a fluent f even if they appear not to be relevant to $\text{propagate}(f)$. The reason for this is that subsequent developments may change this. In the example we have been considering, perhaps proving f has been reduced (after considerable effort) to proving g and h ; when our focus changes to that of proving $\neg f$, we should retain this information in case (as happens in this example) we decide that we need to prove f after all.

4 AN EXAMPLE

The ideas that we have described have been implemented using the multivalued theorem prover described in [Ginsberg, 1989, Ginsberg, 1991b], which allows the user to select truth values from arbitrary bilattices and to include arbitrary modal operators in a declarative database. The precise axiomatization of actions generally and our domain in particular can be found in the appendix.

The domain we are considering involves a gun, which may or may not be loaded, and a victim (Fred), who may or may not be alive. At time 0, the gun is loaded and Fred is alive.

This domain has three actions: loading and unloading the gun (which take a single unit of time and always succeed), and shooting the gun at Fred. If the gun is loaded, firing it at Fred will kill him after a delay of two units of time.⁵ All of the fluents persist in a nonmonotonic fashion except that once Fred dies, he is guaranteed to stay dead.

The course of events in this domain is as follows:

Time	Event
0	Fred is alive and the gun is loaded
1	The gun is unloaded
2	The shooting is attempted and the gun is reloaded
50	The shooting is attempted
52	The gun is reloaded
53	The shooting is attempted

The first shooting action should fail, since the gun has been unloaded at the previous time point. The second shooting action should succeed by default, since the gun has presumably remained loaded between times 3 and 50. The third shooting action will definitely succeed, since it immediately follows a load action.

Given this information, the system was asked to investigate the truth or falsity of the fluent alive (is Fred alive?) at all times; the results are shown in Figures 5

and 6.

The machine reasoned as follows:

- a. Initially, it had no explicit information concerning the state of Fred's health.
- b. The machine first realized that the shooting at time 50 was expected to succeed, so that at times 52 and subsequently, Fred was expected to be dead.
- c. The machine also decided that the shooting at time 2 was likely to succeed, since the gun was loaded in the initial situation and that fact was expected to persist.
- d. The machine next investigated the modal expression $\text{propagate}(\text{loaded})$ a bit more closely, since this modal expression was used in its conclusion that the shootings at times 2 and 50 succeed. It discovered that the gun is not expected to be loaded at time 2 after all, and therefore that the first shooting should fail.
- e. The machine next realized that the shooting at time 53 was guaranteed to succeed, so that at times 55 and subsequently, Fred would certainly be dead.
- f. The previous proof attempts involved attempts to find times at which Fred was expected to be dead; the machine now changed its focus to trying to prove that he was alive, and concluded that he was (by default) from time 0 until the success of the shooting at time 50.
- g. Finally, the program realized that Fred was guaranteed to be alive in the initial situation, and updated its conclusions to reflect this.

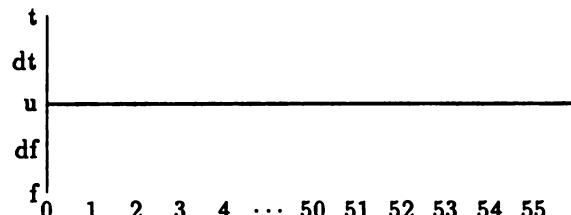
In this example, we can clearly see the two features that have been the focus of this paper – the event-driven nature of the description, evidenced by the lack of computational effort devoted to the “idle” time from $t = 4$ to $t = 50$, and the anytime nature of the analysis, shown in the shifting conclusions displayed in Figures 5 and 6.

5 CONCLUSION

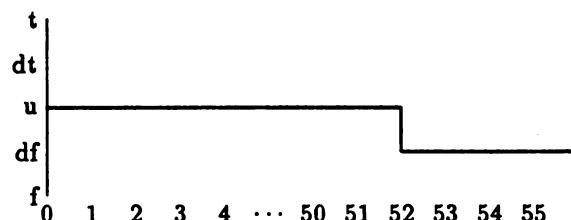
My intention in this paper has been to argue for two things: First, the use of truth values that directly capture the complete history of fluents that change over time; second, the manipulation of these truth values using truth-functional modal operators. We have presented an implementation of our ideas that correctly analyzes a simple example similar to the shooting scenario presented in [Hanks and McDermott, 1987], but the theoretical justifications for this approach are more compelling:

1. The approach embeds the action descriptions in a full declarative language, and investigates the consequences of actions by proving theorems against

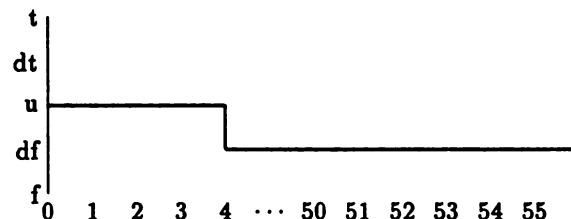
⁵This is the only consequence of the shooting action. Specifically, shooting does not cause the gun to become unloaded.



(a) Initial knowledge



(b) The second shooting succeeds by default



(c) The first shooting also succeeds by default

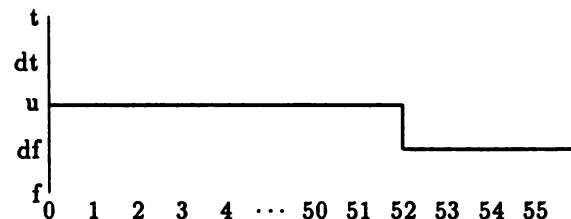
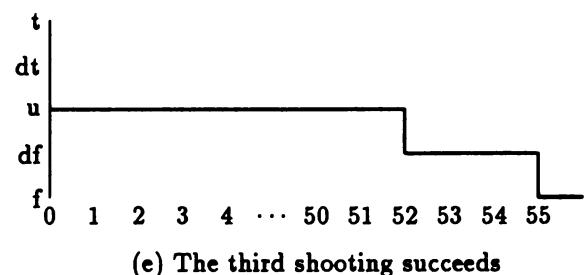
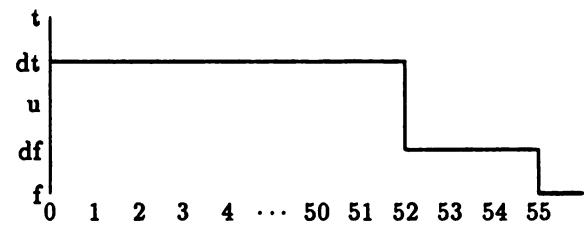
(d) Unload at $t = 1$ causes first shooting to fail

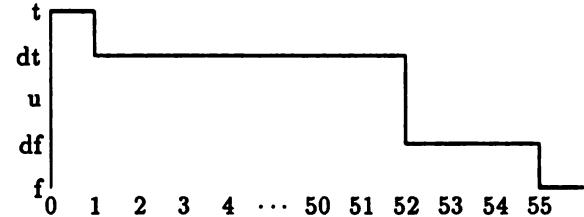
Figure 5: The shooting scenario



(e) The third shooting succeeds



(f) Fred is alive by default initially



(g) In the initial situation, Fred is alive for sure

Figure 6: The shooting scenario (ctd.)

- this background. A system developed in this way will benefit from developments elsewhere in the theorem-proving community in a way that a more *ad hoc* approach cannot.
2. The event-driven nature of the approach allows us to reason in a computationally viable way about fluents that change value only infrequently. As discussed in Section 2.1, we do this without committing ourselves to any specific ontology of time or of action.
 3. The natural implementation of our ideas exhibits an anytime behavior that we can expect to be present in the associated planning system as well. Furthermore, this ability to incrementally refine our conclusions is grounded in a solid formal foundation.
 4. Finally, the approach we have described avoids the use for a blanket frame axiom such as (1), which is likely to fall prey to the problem of induction. Although we have presented no solution to this difficulty, our approach makes clear that such a solution will need to be reflected in the declarative description of our domain, since the expected future history of any particular fluent needs to be asserted when the fluent itself is added to the database.

APPENDIX: AXIOMS USED IN SECTION 4

The axioms used in our description of action involve four separate causal predicates, as follows:

1. *causes(a, f)* means that the action *a* causes the fluent *f* to be true instantaneously.
2. *causes-persistently(a, f)* means that the action causes the fluent to be true in a way that is expected to persist into the future.
3. *causes-forever(a, f)* means that the action causes the fluent to be true in a way that is *guaranteed* to be true in the future.
4. Finally, *causes-not(f₁, f₂)* means that fluent *f₁* implies the negation of fluent *f₂*. The fluents *alive* and *dead* are related in this fashion.

Fluents are reified using a *holds* predicate; the insertion of the reified fluents into the database is also reified using a *triggers* predicate, so that *holds* is the result of applying the modal operator *propagate* to *triggers*.

Here are the axioms associated with these predicates, expressed in a PROLOG-like style:

```
holds(F) :- causes(A,F), duration(A,T),
           delay(succeeds(A),T).
causes(A,F) :- causes-persistently(A,F).
holds(F) :- propagate(triggers(F)).
```

```
triggers(F) :- causes-persistently(A,F),
              duration(A,T),
              delay(succeeds(A),T),
              default.
triggers(F) :- causes-forever(A,F),
              duration(A,T)
              delay(succeeds(A),T).
not(holds(P)) :- causes-not(Q,P), holds(Q).
not(triggers(P)) :- causes-not(Q,P),
                   triggers(Q).
```

There are a couple of things to note here:

1. The MVL system has a true negation operator in addition to the modal negation-as-failure operator used in PROLOG. The *not* appearing in the heads of the last two of the above rules is true negation.
2. The sentence *default* is inserted into the database with a truth value indicating that it has value *dt* at all times. This allows us to use a monotonic *propagate* operator, with the inclusion of the *default* sentence serving to distinguish *causes-persistently* from *causes-forever*.

We also need axioms expressing conditions under which an action succeeds. We assume that there is a predicate *prec(a, p)* that holds whenever *p* is a precondition of the action *a*.

```
succeeds(A) :-
  occurs(A),
  forall(P, (holds(P) :- L (prec(A,P))))).
```

Note the appearance of the quantifier *forall* in the above rule, together with the modal operator of knowledge *L*. This rule says that the action *A* succeeds if it occurs and if every known precondition of *A* holds. The MVL system includes a modal operator of knowledge *L* and true quantifiers.

Finally, there is a dummy action *init* that takes place at time 0 and is used to construct the initial situation. This action has no preconditions and takes no time.

```
occurs(init). ;true at time 0 only
duration(init,0).
```

To describe the shooting domain specifically, we first describe the initial situation as a consequence of the *init* action:

```
causes-persistently(init,alive).
causes-persistently(init,loaded).
```

These axioms say that Fred is alive (and expected to remain so) and that the gun is loaded (and also expected to remain so) in the initial situation.

We also need axioms describing the various actions. Here are *load* and *unload*:

```

causes-persistently(load,loaded).
causes-persistently(unload,unloaded).
duration(load,1).
duration(unload,1).

```

The fluents load and unload are negations of one another:

```

causes-not.loaded,unloaded).
causes-not(unloaded,loaded).

```

Shooting is similar. It causes Fred to be dead forever, has a precondition of the gun being loaded, and needs two units of time to take effect. The fluents alive and dead are negations of one another:

```

causes-forever(shoot,dead).
prec(shoot,loaded).
duration(shoot,2).
causes-not(alive,dead).
causes-not(dead,alive).

```

Finally, we need axioms saying what occurs when:

```

occurs(unload). ; true at time 1
occurs(shoot). ; true at times 2, 50
; and 53
occurs(load). ; true at times 2 and 52

```

When we indicate "true at time t " above, we mean that these facts are inserted into the database with truth values indicating that they are true at these times; there is no way to represent this using conventional PROLOG syntax. The three occurrences of the shooting action are combined to get a truth function similar to that shown in Figure 1 except for the fact that the occurrences are all of unit time duration.

Acknowledgement

This work has been supported by the Air Force Office of Scientific Research under grant number 90-0363 and by NSF under grant number IRI89-12188. I would like to thank Adnan Darwiche, Don Geddis, Drew McDermott and David Smith for various helpful discussions.

References

- [Baker and Ginsberg, 1989] Andrew B. Baker and Matthew L. Ginsberg. Temporal projection and explanation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 906-911, 1989.
- [de Kleer, 1986] Johan de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28:127-162, 1986.
- [Dean and Boddy, 1988] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 49-54, 1988.
- [Dean and Kanazawa, 1988] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 524-528, 1988.
- [Drummond, 1988] Mark Drummond. Situated control rules. Technical report, NASA Ames Research Center, Moffett Field, CA, 1988.
- [Fikes and Nilsson, 1971] R.E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189-208, 1971.
- [Finger, 1987] Jeffrey J. Finger. *Exploiting Constraints in Design Synthesis*. PhD thesis, Stanford University, Stanford, CA, 1987.
- [Genesereth and Nilsson, 1987] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.
- [Ginsberg and Smith, 1988] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35:165-195, 1988.
- [Ginsberg, 1988] Matthew L. Ginsberg. Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265-316, 1988.
- [Ginsberg, 1989] Matthew L. Ginsberg. User's guide to the MVL system. Technical report, Stanford University, 1989.
- [Ginsberg, 1990a] Matthew L. Ginsberg. Bilattices and modal operators. *Journal of Logic and Computation*, 1, 1990.
- [Ginsberg, 1990b] Matthew L. Ginsberg. The computational value of nonmonotonic reasoning. In *Proceedings 1990 Workshop on Nonmonotonic Reasoning*, Lake Tahoe, CA, 1990. American Association for Artificial Intelligence.
- [Ginsberg, 1991a] Matthew L. Ginsberg. Anytime declarativism. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991. Submitted.
- [Ginsberg, 1991b] Matthew L. Ginsberg. The MVL theorem proving system. In *Proceedings of the AAAI Symposium on Implemented Knowledge Representation and Reasoning Systems*, Stanford, California, 1991.
- [Ginsberg, 1991c] Matthew L. Ginsberg. What is the modal truth criterion? In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991. Submitted.
- [Green, 1969] C. C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 4*, pages 183-205. American Elsevier, New York, 1969.
- [Haas, 1987] A.R. Haas. The case for domain-specific frame axioms. In Frank M. Brown, editor, *The*

Frame Problem in Artificial Intelligence. Morgan Kaufmann, San Mateo, CA, 1987.

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logics and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[Kripke, 1971] Saul A. Kripke. Semantical considerations on modal logic. In L. Linsky, editor, *Reference and Modality*, pages 63–72. Oxford University Press, London, 1971.

[Lifschitz, 1986] Vladimir Lifschitz. On the semantics of STRIPS. In *Proceedings of the 1986 Workshop on Planning and Reasoning about Action*, Timberline, Oregon, 1986.

[Myers and Smith, 1988] Karen L. Myers and David E. Smith. On the persistence of derived beliefs. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988.

[Schubert, 1990] Lenhart K. Schubert. Monotonic solution of the frame problem in the situation calculus. In Henry E. Kyburg, Jr., Ronald P. Loui, and Greg N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, Boston, 1990.

[Skyrms, 1966] Brian Skyrms. *Choice and Chance: An Introduction to Inductive Logic*. Dickerson, 1966.

[Wilkins, 1988] David E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, San Mateo, CA, 1988.

The Computational Value of Nonmonotonic Reasoning

Matthew L. Ginsberg
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

A substantial portion of the formal work in artificial intelligence over the past decade has involved the formalization of nonmonotonic reasoning. Unfortunately, although much of the original motivation for this work argued that it was necessary for intelligent systems to jump to default conclusions quickly, little of the formal work has justified these arguments: Existing formal theories of nonmonotonic reasoning are nonsemidecidable, with implementations involving repeated calls to theorem provers.

We suggest that the computational value of nonmonotonic reasoning lies in its ability to focus the inference process in a hierarchical way. We show that a formal system capable of drawing tentative conclusions *without* performing a consistency check is capable both of jumping to default conclusions and, perhaps more significantly, of achieving substantial computational savings when a nondefault conclusion is required.

1 INTRODUCTION

In all of the applications that have been proposed for nonmonotonic reasoning, it is possible to draw default conclusions quickly; the computational expense of the default approach comes from the need for some sort of consistency check. In most cases, it is necessary to check that the default conclusions are consistent with the initial database.

A default approach to reasoning about action, for example, requires two invocations of a theorem prover in order to determine whether or not some domain fact is preserved when an action is executed. In the first, we simply use the frame axiom to conclude that it is preserved; in the second, we check the consistency of this conclusion.

Here, the first "proof" consists simply of an application of the frame axiom; the work is in the consistency check. Since a monotonic approach to reasoning about action involves checking a variety of persistence conditions in order to determine that the domain fact persists through the action, it is possible for the nonmonotonic approach to be the more efficient of the two. This argument is made quantitative in [Ginsberg and Smith, 1988].

Even though the nonmonotonic approach requires an additional call to a theorem prover, it is possible for it to be more efficient than the conventional one because each of the two proof attempts may be much more efficient than its monotonic counterpart. In this paper, we suggest that this is the true role of nonmonotonic reasoning in artificial intelligence: the replacement of one complex proof with two or more simpler ones.

The computational benefits of this sort of approach are well known. Consider the basic idea underlying island-driven search [Erman and Lesser, 1975] or hierarchical planning [Sacerdoti, 1977]. By identifying an island through which a plan or search must pass, it is possible to split one search problem into two smaller ones; since the difficulty of the problem grows exponentially with its size, this leads to computational savings.

In general, it is possible that we are not *certain* that the solution to a given search problem passes through some specific island, but only *expect* it. In this case, we search first for a solution passing through the proposed island, and subsequently for one that avoids it. Provided that the cost of identifying the hypothetical island is outweighed by the expected gain in solving the smaller search problem, the identification of the island is a justifiable expense.

As we will see, the identification of the island corresponds to the discovery of a default solution to the problem, and the subsequent consistency check has an analog in the search for a solution that passes through the island in question. This allows us to rephrase the observation in the previous paragraph as follows: Provided that the cost of the first default proof is out-

weighed by the expected gain in checking the consistency of this default proof as opposed to solving the original problem, the use of default information is a justifiable expense.

In the next section, we will discuss this issue in terms of an extended example involving the use of an ATMS to solve a synthesis problem. A discussion of the computational advantages offered by this idea is contained in Section 3. Planning examples are discussed in Section 4, and an examination of the nature of the "consistency check" required by existing nonmonotonic formalisms is the topic of Section 5. Connections with existing work on search control and hierarchical problem solving are examined in Section 6.

2 SYNTHESIS PROBLEMS

The focus of this paper will be on synthesis problems, such as planning or circuit design. As discussed in [Green, 1969], one way in which to implement a planning system is to use resolution to prove that there exists a situation in which the goal has been achieved and to then recover the actions needed to achieve the goal by examining the situation so constructed.

[Finger, 1987] hints at another possible approach. Here, the idea is that we have some (probably inconsistent) set A of assumptions, and are looking for a consistent subset S of A that allows us to conclude that the goal has been achieved. Our assumption set A might consist of a variety of assumptions about which actions were taken at what times; if A includes both the assumption that we crossed the street at time t and the assumption that we remained stationary at that time, A itself will be inconsistent and we will need to commit to at most one of these actions when actually constructing our plan.

Of course, finding a consistent subset of A that entails some specific goal g is what ATMS's are all about [Reiter and de Kleer, 1987]. (The restriction that S above be consistent is the same as de Kleer's condition that S not contain a nogood [de Kleer, 1986].) We will refer to this type of problem in general as follows:

Definition 2.1 Suppose that we have fixed a base theory T , a set of assumptions A and a goal g . We will call the problem of finding a subset S of A that is consistent with T such that $T \cup S \models g$ the (T, A, g) synthesis problem.

The thrust of this paper is that we are able to use default assumptions to focus this process. Thus in planning a trip from Stanford to MIT, I may generate the incomplete plan of getting to San Francisco airport (SFO) and thence to MIT via Logan; I am assuming that the intermediate steps of getting from Stanford to SFO, from SFO to Logan, and from Logan to MIT are all things that I can accomplish if I consider the

problem in more detail.

Let us suppose for the moment that we are actually in a position to prove rigorously that we can get from Stanford to MIT via the airports suggested. (In practice, of course, all sorts of things can go wrong, and we must leave many small details of the trip unplanned until we actually encounter them.) If this is the case, then our decision to go via SFO and Logan really has done little more than focus our search effort – the detailed plan could have been generated initially instead.

What we have done is to augment our assumption set A with a set of default assumptions D (in this example, the assumptions that we can get from Stanford to SFO and so on). Then, having solved the planning problem given the default assumptions, we attempt to eliminate these assumptions by showing that they are in fact provable from our original set A after all. Here, then, is the procedure for solving the (T, A, g) synthesis problem using the default set D :

Procedure 2.2 To solve the (T, A, g) synthesis problem:

1. Solve the $(T, A \cup D, g)$ synthesis problem to obtain a solution $\{a_1, \dots, a_m, d_1, \dots, d_n\}$.
2. Set $A_0 = \{a_i\}$ and $T_0 = T \cup A_0$ and solve the (T_0, A, d_1) problem to obtain A_1 .
3. Now set $T_1 = T_0 \cup A_1$ and solve (T_1, A, d_2) to obtain A_2 . Repeat to eventually solve (T_{n-1}, A, d_n) to obtain A_n .
4. Return $\cup A_i$ as a solution to the original problem.

This procedure works by first solving the original problem using the defaults, and then replacing the defaults one at a time by sets of nondefault assumptions that entail them. As we have described it, we are assuming that it is always possible to replace the defaults in this fashion, so that they are all consequences of information elsewhere in the database; in practice, this may not be the case. In the next section, we will modify the above procedure to cater to this possible difficulty.

Before proceeding, however, I should point out that the idea of using defaults to represent assumptions involved in solving synthesis problems is not a new one. The connection between ATMS's and nonmonotonic reasoning is well-understood and has been described by a variety of authors [Ginsberg, 1989, Reiter and de Kleer, 1987, and others]; the THEORIST system is built around the idea that defaults represent hypotheses that can be used in problem solving [Poole et al., 1985]. The contribution made by this paper is not in identifying the connection between nonmonotonic reasoning and hypothetical solutions to synthesis problems, but in proposing that these hypothetical solutions can be used to focus the search for more certain results.

3 COMPUTATIONAL VALUE

Whether or not using defaults to focus search is a good idea from a computational point of view depends upon a variety of assumptions about the set of defaults D : Can these defaults really be used effectively to shorten the length of the original proof? Are the defaults likely to point the way to a solution to the original synthesis problem, or are they likely to be blind alleys? And so on.

In order to give a quantitative estimate of the time Procedure 2.2 needs to solve a particular query, we will need to make a variety of assumptions about these issues. Specifically, we assume that:

1. The relative cardinalities of the sets D and A is given by λ . We take

$$\lambda = \frac{|D|}{|A|}$$

the ratio of the number of defaults to the number of original assumptions. We will go further and assume that since the number of assumptions in the default synthesis problem $(T, A \cup D, g)$ is $1 + \lambda$ times the number of assumptions in the original synthesis problem (T, A, g) , the branching factor in the new problem is $(1 + \lambda)b$, where b is the branching factor in the original problem.

2. If S is a solution to some synthesis problem with branching factor b and there are m assumptions in S , then the time needed to solve the synthesis problem is of order $b^m = b^{|S|}$. This will be valid if b is the effective branching factor between additions to the set of assumptions being constructed as a solution to the problem.
3. Suppose that S is the smallest solution to a synthesis problem (T, A, g) . Then there is some $n \geq 1$ such that the expected size of the smallest solution to the default synthesis problem $(T, A \cup D, g)$ is $|S|/n$.

Roughly speaking, n here is the number of original assumptions that we expect to be covered by a single default rule; if $n = 1$, we do not expect the defaults to be useful in shortening the length of the solution.

4. There is some fixed l and probability p such that for an arbitrary default $d \in D$ and an arbitrary subset $A' \subseteq A$ consistent with T , p is the probability that the synthesis problem $(T \cup A', A, d)$ admits a solution of length l or less.

We clearly expect $l \geq n$, since each default covers on average n base-level assumptions; if p is relatively large and l relatively small subject to this condition, each default is likely to expand into a fairly small number of nondefault assumptions. In general the probability that the $(T \cup A', A, d)$ synthesis problem admits a solution may vary depending on the sets A and A' and the particular

default d ; it suffices for the probability that this synthesis problem admits a solution to be no less than p .

The procedure that we will consider is a slight variant of Procedure 2.2. Here it is:

Procedure 3.1 *To solve the (T, A, g) synthesis problem:*

1. *Solve the $(T, A \cup D, g)$ synthesis problem to obtain a solution $\{a_1, \dots, a_m, d_1, \dots, d_n\}$.*
2. *Set $A_0 = \{a_i\}$ and $T_0 = T \cup A_0$ and attempt to solve the (T_0, A, d_1) problem to obtain A_1 , requiring that $|A_1| \leq l$.*
3. *If no such A_1 exists, abandon the default solution and solve the original synthesis problem (T, A, g) directly.*
4. *Otherwise, set $T_1 = T_0 \cup A_1$ and attempt to solve (T_1, A, d_2) to obtain A_2 with $|A_2| \leq l$. Repeat to either eventually solve (T_{n-1}, A, d_n) to obtain A_n ; if A_i doesn't exist for some i , solve (T, A, g) directly.*
5. *Return $\cup A_i$ as a solution to the original problem.*

As remarked at the end of the previous section, the reason that we use this modified procedure is that it is possible that the default solution found in step 1 does not expand into a full solution, and we need to bound the amount of time spent trying to expand the various defaults in subsequent steps.

Note also that if step 2 or 4 fails to find a solution using the extended assumption set $A \cup D$, we simply give up, as opposed to either backtracking to another default solution or attempting to modify the existing solution in a way that addresses the difficulty. Even given this extremely conservative behavior, however, Procedure 3.1 leads to computational savings:

Theorem 3.2 *Let (T, A, g) be a synthesis problem for which the shortest solution is of length k . Then provided that $l \ll k$, it will be more efficient to solve this synthesis problem using Procedure 3.1 than by conventional means whenever*

$$p > \frac{1 + \lambda}{b^{n-1}} \quad (1)$$

Proof. First note that for a problem with solution of length k , the length of the solution to the synthesis problem that uses the defaults is expected to be k/n . Since each assumption in this solution (and there are at most k/n of them) has probability at least p of being successfully replaced with l or fewer nondefault assumptions, it follows that the probability that Procedure 3.1 solves the synthesis problem using the defaults is at least $p^{k/n}$. Thus the total time needed to

solve the problem is bounded by

$$[(1 + \lambda)b]^{k/n} + \frac{k}{n}b^l + (1 - p^{k/n})b^k \quad (2)$$

The first two terms here represent costs that are always incurred – that of solving the default synthesis problem with branching factor $(1 + \lambda)b$ and depth k/n and of attempting to replace the at most k/n defaults (each leading to a problem of depth l and branching factor b). The final term reflects the cost of solving the problem without using the defaults; as argued in the previous paragraph, the probability that this cost is incurred is at most $1 - p^{k/n}$.

In order for our ideas to lead to computational savings, the expression in (2) needs to be less than b^k , the expected cost of solving the problem using conventional methods.

Note first that in light of the condition that $l \ll k$, the second term in (2) is dominated by the first and third; what we are arguing here is basically that the problem of replacing the default assumptions is far easier than that of solving the original synthesis problem itself. Thus the condition that we need to satisfy becomes:

$$[(1 + \lambda)b]^{k/n} + (1 - p^{k/n})b^k < b^k$$

or

$$[(1 + \lambda)b]^{k/n} < p^{k/n}b^k \quad (3)$$

Raising both sides to the power n/k gives us

$$(1 + \lambda)b < pb^n$$

which is clearly equivalent to the conclusion of the theorem. \square

Theorem 3.2 is of substantial practical value. Consider, for example, the conservative assumptions that $n = 2$ (so that every default corresponds to only 2 non-default assumptions) and $\lambda = 1$ (so that including the default assumptions doubles the size of the assumption set). Now (1) becomes

$$p > \frac{2}{b}$$

Since most synthesis problems have fairly large branching factors, this is a very weak restriction.

Theorem 3.2 can also be rewritten in the following form:

Corollary 3.3 *Procedure 3.1 can be expected to lead to computational savings whenever*

$$\frac{t(\text{default})}{t(\text{original})} < \text{prob}$$

where $t(\text{default})$ is the expected time needed to solve the default synthesis problem, $t(\text{original})$ is the time needed to solve the original one, and prob is the probability that the default solution can be expanded to a complete one.

Proof. Given the arguments in the proof of Theorem 3.2, this is a straightforward rewriting of (3). \square

Roughly speaking, it is worth risking an “investment” of 10% of the time needed to solve a particular problem if we have more than a 10% chance of solving the problem in this fashion. It is interesting to note that the cost of replacing the defaults in the tentative solution is sufficiently small that it does not enter into the above result.

4 PLANNING EXAMPLES

The idea that default rules can be used to generate islands in a synthesis problem allows us to reinterpret the default information typically used by systems that reason about action. As an example, consider the frame axiom. This tells us that facts true in one situation will remain true in subsequent situations:

$$\text{holds}(p, s) \wedge \neg \text{ab}(a, p, s) \supset \text{holds}(p, \text{result}(a, s)). \quad (4)$$

Applied to planning, (4) suggests that when planning for a conjunctive goal, we simply achieve the subgoals individually. The frame axiom gives us reason to assume that previously achieved subgoals will not be clobbered by subsequent actions.

Of course, this is not valid in general. But much of the computational expense involved in planning is a consequence of the need to regress all of the goals through the various actions being considered. In situations where this regression is not needed, an optimistic approach will succeed much more quickly than the conventional one; gambling that this will indeed be the case is likely to be worthwhile because the payoff is quite large if we are correct. This observation is also made in [Elkan, 1990].

As another example, consider what might be called the “qualification assumption.” By this we mean the assumption commonly made when reasoning about action that actions are unqualified in the absence of information to the contrary. Thus we assume that an attempt to start a car will succeed even if there is no information available explicitly stating that there is no potato in the car’s tailpipe. Perhaps more realistically, attempts to drive the car are assumed to succeed even in the absence of specific information indicating that none of the tires is flat.

Applying this idea to planning leads us to assume that our intended actions are unqualified. This will typically be the case, and we will be saved the effort of worrying about potential qualifications as we construct the plan.

Two other advantages also come to mind here, although we will need to extend Procedure 3.1 in order to exploit them. The first involves cases where although the qualification assumptions hold (i.e., there

is no potato in the tailpipe), we do not know that they hold. In this case, there may well be no solution to the original problem, since the necessary knowledge (about the state of the tailpipe) is simply missing from our database.

This is rather different from the previous ideas that we have considered. From a formal point of view, we are presented with incomplete domain information in the sense that our base set T is only a subset of the "complete" base set T' . Although we really want a solution to the (T', A, g) synthesis problem, we are forced to attempt to solve the (T, A, g) synthesis problem. Using the defaults, we solve the $(T, A \cup D, g)$ synthesis problem instead, and observe:

1. If the defaults used in the solution are all consequences of T' (as is the case with qualification assumptions), then the solution obtained will in fact be a solution to the (T', A, g) synthesis problem.
2. Even if not all of the defaults used are consequences of T' , the solution can still focus our subsequent efforts in a fashion similar to that described in Section 3.

Finally, it may be the case that the qualification assumption actually fails – perhaps there is a potato in the tailpipe after all. In this case, the default solution to the synthesis problem can be expected to be similar to the solution to the problem itself (which presumably involves clearing the tailpipe and then proceeding as planned). Making practical use of this idea will involve considerable work in extending Procedure 3.1, of course.

5 THE CONSISTENCY CHECK

In the synthesis problems we have examined, we proceeded by first finding a default solution to the query, and then filling in the details of this skeletal solution. Thus instead of simply using a default rule to conclude that it is possible to fly from San Francisco to Boston, we can consult with a travel agent and confirm that it is indeed possible, and that American Airlines flies the route in question.

Default inference does not generally proceed in this fashion. Instead, approaches such as that in [Reiter, 1980] function first by finding a default solution to a problem, and then attempting to prove that the default solution *cannot* be extended to a certain one. Instead of finding a flight from San Francisco to Boston, we would attempt to show that there cannot be one. (Perhaps only Boeing 747's can fly transcontinentally, and the runways at Logan are too short for them to land.) When we fail to conclude that our default solution is *invalid*, we assume that it is in fact acceptable.

This is in contrast with the approach we have described, where we first draw a default conclusion and then attempt to prove it to be monotonically valid. But provided that our initial database is consistent, both approaches lead to the same result. Given that we can prove that our default conclusion is in fact valid, we do not need to attempt to prove it false. In our example, the existence of the American flight from San Francisco to Boston guarantees that we will fail in our attempt to prove that no such flight exists.

We have already remarked that in other instances it may be that there is no monotonic proof that we will be able to complete some portion of our skeletal plan. Consider the initial action of driving to the San Francisco airport: The highway may be closed, we may run out of gas, our car may be in the shop, and so on. Here, we may indeed take the alternative tack of considering possible proofs that we *can't* drive to the airport.

If the consistency check succeeds, so that we are unable to prove that there is *no* solution to our original problem that passes through the default island, we tentatively conclude that we will be able to complete the plan at some later date. Thus the final leg of our journey is left almost completely unspecified – I may have some vague notion that I'll rent a car when I get into Boston, but no idea about which car rental agency I'll use, how I'll find them, and so on. This is similar to the discussion of the qualification problem in the last section, where we conceded that it might be possible to find a solution to a synthesis problem given an incomplete description of the domain in question.

The consistency check may fail, however. Perhaps my car is in the shop and I can't drive to San Francisco airport or, as in the previous section, perhaps there is a potato in my tailpipe after all. Procedure 3.1 would now have us simply abandon our default plan and construct a monotonic plan after all. More effective would obviously be to "debug" the default solution to the original synthesis problem. Although there has been no formal work on this sort of a process, the approach we have outlined may be a useful first step in this direction. Again, we saw an example of this in the previous section.

Before turning to other issues, we should remark that there is another way in which we can understand our ideas generally. The reason that most approaches to nonmonotonic reasoning require a consistency check is that a declarative system is useless without it – there is simply no point in asserting a conclusion that is provably false.

In control problems, this is not the case. If it is possible to draw quick conclusions about the portion of a search space that should be expanded next, those conclusions have value even if subsequent consideration would show them to be false. The reason is that the potentially large time needed for this "subsequent con-

sideration" may not be justified in terms of the time saved in the search itself. "Fast but not accurate" is unacceptable behavior for most declarative systems, but is *necessary* behavior in control work; because we are using the results of our default analysis primarily to control subsequent search, we are able to take advantage of default conclusions without explicitly checking their consistency.

6 RELATIONSHIP TO EXISTING WORK

6.1 HEURISTIC SEARCH AND METALEVEL ACTIVITY

One way to think of the ideas we have presented is as describing a connection between default reasoning and the control of search. The basic suggestion we have made is that an important use of nonmonotonic reasoning in AI is as a means of focussing search when considering a difficult problem. Previous investigations of the problem of search control have relied either on the use of heuristic rules or domain-independent techniques developed by considering statistical information about the contents of the database [Smith and Genesereth, 1985].

By using default information to guide search, we are applying a domain-independent heuristic to domain-dependent default information. The heuristic is to apply the default rules to narrow the scope of the problem; the domain-dependent information is in the form of default rules such as the frame axiom. The advantage of using the information in this fashion is that it simplifies the knowledge engineering problem to some extent: All of the domain-dependent information being used is in the form of declarative information about the domain being considered. In many instances, this declarative information will already be present when the synthesis problem is confronted - we can, for example, expect our planning system to already be equipped with some representation of the frame axiom. The idea that one should control search by applying domain-independent control rules to domain-dependent base-level information and not by a direct appeal to domain-dependent control rules also appears in [Ginsberg and Geddiss, 1991].

There is a close connection between our ideas and those appearing in [Smith and Genesereth, 1985], which suggests examining a knowledge base to determine an ordering for the clauses in a conjunctive query. There is a clear connection between the "average number of solutions to a conjunct" [Smith and Genesereth, 1985, p. 180] and the validity of a default rule such as the frame axiom. It should be noted, however, that the goals of [Smith and Genesereth, 1985] are different from ours: The earlier work orders the conjunctive goals in the

problem as stated, while we are introducing new ones (the islands) based on default information.

The approach we have proposed also makes it possible to draw default conclusions about the domain itself. Even if no travel agent is available, it is important to be able to *surmise* that it will be possible to fly from San Francisco to Boston. This point is even clearer for portions of a skeletal plan (such as driving to the airport) that cannot be expanded until the plan is executed. The argument that default information is needed by autonomous agents functioning in uncertain or unpredictable domains is hardly a new one; we have merely formalized the old idea that this same information can be used in controlling search as well.

6.2 HIERARCHICAL PLANNING AND DESIGN

The approach we have described is little more than a formalization of the conventional notion of hierarchical problem solving. Nevertheless, we feel that the contribution made by this paper is important for two separate reasons.

First, it provides a general description of the idea of hierarchical problem solving. This description is independent of whether the application is to planning, design or some other synthesis problem.

Far more importantly, the information used to describe the planning (or other) hierarchy is declarative. Multiple levels of hierarchy can clearly be handled using prioritized defaults; the key point here is that the split is not an *ad hoc* one introduced by the system designer, but instead a natural one generated from existing declarative information.

It might seem that this is a small point, and that all of the usual problems encountered in hierarchical problem solving (e.g., the fact that potential islands need to be identified by the system designer) remain in our approach, but this is not the case. As an example, consider just this difficulty - where do the defaults come from? We can offer the following suggestions:

1. They can be entered by the system designer as usual, with his intention being to control search. Our language is somewhat different, but no expressive power has been lost.
2. They can be part of the declarative knowledge base for another reason. An example is the frame axiom, which will surely be part of any system that plans and reasons about action. Although not made a part of the system for control reasons, it appears that the frame axiom can be used effectively in this fashion.
3. They can be learned from previous problem solving efforts. Caching partial plans can now be viewed declaratively as a matter of caching the

defaults indicating that these plans will be useful.

Another important feature of the declarative approach we have proposed is that it may allow us to understand the process of plan debugging. As an example, consider a hierarchical planner that has failed to expand a particular high-level plan into lower-level actions. If the planning hierarchy is described procedurally, there is no real way for the planner to overcome its difficulties. But if the planning hierarchy is a consequence of declarative default information, it will be possible to trace the failure to the failure of a specific default rule – and that may well be a necessary first step toward salvaging the high-level plan in some way.

Acknowledgements

This work has been supported by NSF under grant number DCR-8620059 and by the Rockwell Palo Alto laboratory. I would like to thank Karen Myers, Narinder Singh and David Smith for many useful discussions and suggestions.

References

- [de Kleer, 1986] Johan de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28:127–162, 1986.
- [Elkan, 1990] Charles Elkan. Incremental, approximate planning. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 145–150, 1990.
- [Erman and Lesser, 1975] Lee Erman and Victor Lesser. A multi-level organization for problem-solving using many diverse, cooperating sources of knowledge. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pages 483–490, Tbilisi, USSR, August 1975.
- [Finger, 1987] Jeffrey J. Finger. *Exploiting Constraints in Design Synthesis*. PhD thesis, Stanford University, Stanford, CA, 1987.
- [Ginsberg and Geddis, 1991] Matthew L. Ginsberg and Donald F. Geddis. Is there any need for domain-dependent control information? In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991. Submitted.
- [Ginsberg and Smith, 1988] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35:165–195, 1988.
- [Ginsberg, 1989] Matthew L. Ginsberg. A circumscriptive theorem prover. *Artificial Intelligence*, 39:209–230, 1989.
- [Green, 1969] C. C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 4*, pages 183–205. American Elsevier, New York, 1969.
- [Poole *et al.*, 1985] D. Poole, R. Aleliunas, and R. Goebel. THEORIST: A logical reasoning system for defaults and diagnosis. Technical report, University of Waterloo, 1985.
- [Reiter and de Kleer, 1987] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 183–188, 1987.
- [Reiter, 1980] Ray Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
- [Sacerdoti, 1977] Earl D. Sacerdoti. *A Structure for Plans and Behavior*. American Elsevier, New York, 1977.
- [Smith and Genesereth, 1985] David E. Smith and Michael R. Genesereth. Ordering conjunctive queries. *Artificial Intelligence*, 26(2):171–215, 1985.

Updates and Counterfactuals

Gösta Grahne*
 Computer Systems Research Institute
 University of Toronto
 Toronto, Canada M5S 1A4
gosta@db.toronto.edu

Abstract

We study the problem of combining *updates*—a special instance of theory change—and *counterfactual conditionals* in propositional knowledgebases. Intuitively, an update means that the world described by the knowledgebase has changed. This is opposed to *revisions*—another instance of theory change—where our knowledge about a static world changes. A counterfactual implication is a statement of the form ‘If *A* were the case, then *B* would also be the case’, where the negation of *A* may be derivable from our current knowledge. We present a decidable logic, called **VCU**², that has both update and counterfactual implication as connectives in the object language. Our update operator is a generalization of operators previously proposed and studied in the literature. We show that our operator satisfies certain postulates set forth for any reasonable update. The logic **VCU**² is an extension of D. K. Lewis’ logic **VCU** for counterfactual conditionals. The semantics of **VCU**² is that of a multimodal propositional calculus, and is based on possible worlds. The infamous *Ramsey Rule* becomes a derivation rule in our sound and complete axiomatization. We then show that *Gärdenfors’ Triviality Theorem*, about the impossibility to combine theory change and counterfactual conditionals via the Ramsey Rule, does not hold in our logic. It is thus seen that the Triviality Theorem applies only to revision operators, not to updates.

‘Kangaroos have no tails.’

1 BACKGROUND

The usual *material implication* of formal logic fails to capture many implicative statements of natural language. Consider for instance the sentence ‘*If I had some oars, I could row across the river*’. If we find ourselves on the bank of a river, equipped with a boat but no oars, then according to the classical truth-table semantics the above sentence is true, in accordance with our intuition. But then, the truth-table method also claims that the sentence ‘*If I had some oars, pigs could fly*’ is true, which obviously contradicts our intuition (cf. Ginsberg, 1986).

Much work in logic has been devoted to *conditionals*, that is, implications other than the material one. (For an overview of the field, see (Nute, 1984).) A *counterfactual conditional*, or *counterfactual* for short, is a statement of the form ‘*if *A*, then *B**’, where, as the name indicates, the premise *A* can contradict the current state of affairs, or our current knowledge.

The application of counterfactuals to knowledgebases lies in their ability to express rules of the form ‘*If *A*, then *B**’, and questions of the form ‘*What if *A*?*’, where the negation of *A* may be derivable from the knowledgebase. Among others, Bonner (1988) studies a database query language with a *What if*-capability. Ginsberg (1986) describes a large number of AI applications of counterfactuals.

So far, we have looked at one side of the coin. On the other side there is the problem of changing the information represented in a knowledgebase according to new facts, including facts that contradict the information already in the knowledgebase. For a simple example, suppose that the knowledgebase contains the fact that ‘*Tweety is a bird*’ and the rule ‘*All birds can fly*’. Then the newly acquired fact ‘*Tweety cannot fly*’ contradicts the knowledgebase. Thus we have to give up something. A radical solution would be to throw out all the old information from the knowledge-

*On leave from University of Helsinki, Department of Computer Science, Teollisuuskatu 23, SF-00510 Helsinki.

base, and take the new fact as the new knowledgebase. There is though a quite substantial body of work on this problem, both in the fields of AI and databases, as well as in the field of logic. (For some overviews, the reader should turn to (Katsuno & Mendelzon, 1990), (Winslett, in press), and (Makinson, 1985).) A generic name for the problem is *theory change*. The agreement on theory change is that in the case of contradictions, ‘as little as possible’ of the old theory should be changed in order to accommodate the new facts, that is, the change should be minimal. In the example above, minimal change could mean that we give up either the fact ‘*Tweety is a bird*’, or the rule ‘*All birds can fly*’, but not both.

The notion of minimal change also plays a role in defining the meaning of counterfactual implications. This is explicitly expressed in the *Ramsey Rule* (Ramsey, 1950). The Ramsey Rule is summarized by Gärdenfors (1986) as follows:

Accept a proposition of the form ‘if *A*, then *B*’ in a state of belief [or knowledge] *K*, if and only if the minimal change of *K* needed to accept *A*, also requires accepting *B*.

Gärdenfors (1986, 1988) has shown that with certain assumptions on the change operation, such operations are incompatible with the Ramsey Rule, meaning that any logic containing the two is trivial (in a sense to be defined later). In other words, it seems to be impossible to combine the facility to change the knowledgebase with the ability to express *What if*-questions, and have counterfactual *If-then*-rules, while keeping the very intuitive Ramsey Rule. This result is known as the *Gärdenfors Triviality Theorem*.

Many solutions have been sought to the dilemma of the theorem. Gärdenfors (1988) considers weaker versions of the Ramsey Rule, but concludes that these versions do not satisfy our intuition. A more promising direction, where counterfactual *If-then*-rules are forbidden in the knowledgebase, is taken by Levi (1988). By going in this direction we however lose the expressive power of the knowledgebase, as well as the power to express *iterated* counterfactuals, that is rules of the form *If (If A, then B), then C*. On the other hand, Rott (1986) has tried to forbid iterated counterfactuals while allowing (non-iterated) counterfactuals in the knowledgebase, and concluded that the triviality result still stands. The works by Makinson (1990) and Cross (1989) consider the case where the inference operator is non-monotonic, instead of the classical one, but Makinson shows that the triviality theorem holds for all reasonable inference operators. Finally, Gärdenfors (1988) and Arlo Costa (1989) show that weakening the assumptions on the change operator does not provide a way out of the dilemma.

As pointed out in e.g. (Abiteboul & Grahne, 1985), (Katsuno & Mendelzon, 1991), and (Keller &

Winslett, 1985), there are however two fundamentally different ways of changing knowledge. One can regard the new evidence as contributing to our knowledge about the real world (e.g. *the butler is guilty*). On the other hand, the evidence can reflect a change in the real world (e.g. *the master of the house has been murdered*). Changes of the former kind are called *revisions* by Katsuno and Mendelzon (1991), whereas the latter type is given the name *updates* in that same paper. Gärdenfors and his colleagues have set forth a set of rationality postulates for revision operations (Alchourrón, Gärdenfors & Makinson, 1985). Katsuno and Mendelzon (1991) have done the same for update operations. Now Gärdenfors’ triviality result holds for the case where the change function satisfies certain (weakened) postulates for revision.

In this paper we study the problem of combining updates (as opposed to revisions) and counterfactuals via the Ramsey Rule. Our formalism is that of a multimodal propositional logic. Neither operation is taken as more primitive than the other, and both are connectives in the object language (in contrast to e.g. (Gärdenfors, 1988), where the revision operator appears on the level of models). The interpretation of the update connective is a generalization of a ‘possible models approach’ previously proposed and studied in the literature. The interpretation of the counterfactual connective is the one associated with D. K. Lewis’ logic VCU (Lewis, 1973).¹ It turns out that the logic characterized by this combined class of interpretations is obtained by adding the Ramsey Rule as a *derivation rule* to an axiomatization of VCU. The resulting logic, called VCU², of counterfactuals and updates is decidable. We also show that the update postulates of Katsuno and Mendelzon (1991) are theorems or metatheorems in our logic, and that the Gärdenfors Triviality Theorem does not hold in VCU².

2 THE LOGIC VCU²

The Language $\mathcal{L}_{>,o}$

Let \mathcal{L} be the language of propositional calculus, i.e. $\mathcal{L} = \{p_i : i \in \omega\} \cup \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \perp, \top, (\), \}\$. The symbols \perp and \top are 0-ary connectives, and denote the constants *false* and *true*, respectively. The set of all well formed sentences of \mathcal{L} is defined in the usual way. Parentheses are omitted wherever there is no risk of confusion. The language $\mathcal{L}_{>,o}$ is obtained by adding the two (non truth-functional) binary connectives $>$ and \circ to \mathcal{L} . This induces the set of all well-formed sentences of $\mathcal{L}_{>,o}$. When there is no risk of confusion, we shall by a *sentence* mean a well-formed $\mathcal{L}_{>,o}$ -sentence.

¹VCU stands for *variably strict conditionals* with universality. Strict conditionals are of the form *necessarily p implies q*. Universality means that all worlds are accessible from all other worlds.

Letters like ϕ and ψ will be used for sentences.

Intuitively, $\phi \circ \psi$ means the ‘the result of *updating* ϕ with ψ ’. The intuitive meaning of $\phi > \psi$ is ‘If ϕ were true, then ψ would also be true’.

In a knowledgebase setting the language can be used as follows: Let KB be a finite set of $\mathcal{L}_{>,o}$ -sentences. If we want to update the knowledgebase with some new information represented by a $\mathcal{L}_{>,o}$ -sentence ϕ , the new knowledgebase will be represented by the sentence $KB' \circ \phi$, where KB' is the conjunction of the sentences in KB . Note that nothing (except perhaps bounded computational resources) prevents neither the knowledgebase, nor the sentence ϕ to contain the \circ and $>$ connectives in an arbitrary (well-formed) combination. Likewise, a hypothetical query, such as ‘*would* ψ be true, if ϕ were true?’ is evaluated by testing whether KB logically implies $\phi > \psi$.

Changing Worlds and Theories

Usually a KB is *incomplete*, in the sense that the reality it describes allows for several (non-isomorphic) interpretations. That is, the ‘*true real world*’ is known to be among a finitely describable set of possibilities. Which one of these idealized images it is, we do not know.² Now, when a change—intended or not—occurs in the real world, we must change our description of the world. The change itself happens to, or in, the real world. But since we are confined to our set of possibilities, we must perform the ‘thought experiment’ of separately having the change come true in all of our candidate worlds. Each one of these thought-of-worlds, changes ‘*as little as possible*’ in order to accommodate itself to the new order of things. By doing so, it has in fact become another idealized image, one that is not too far from the old one, according to some measure. Our new description of reality should now correctly reflect the outcome of this ‘thought experiment’. The function that maps the old description to the new, is called an *update*.

The approach described above is also known as the ‘possible models approach’ (Winslett, in press):

In updating a theory T with a sentence ϕ , update each model of T separately. For each model M of T , choose those models of ϕ that are closest to M . The result is then the theory of the union of all separate updates.

The scenario is different in the case where we have gathered some new information of what we know is and is not possible in a static world. Our revised description of the real world will then have to take this into account, in that the set it now describes either contains less (*expansion* of knowledge) or more (*contraction* of

²In the words of J. Hintikka (1962), ‘one has to keep an eye on more than one possible world’.

knowledge) possible worlds. The most intriguing case in this scenario is that of a *revision*: What should become of our description if newly acquired information completely contradicts what we thought were among the possibilities of the truth?

$\mathcal{L}_{>,o}$ -Models

Before going into the formal definitions of models, we shall give a—hopefully intuitive—rudimentary account of the interpretation of sentences in $\mathcal{L}_{>,o}$.

We consider a set I of possible worlds. If a sentence ϕ is true in a world $i \in I$, we say that i is a ϕ -world. To capture the concept of ‘as little as possible’, we will associate a total pre-order \leq_i over I with each possible world i . Then $j \leq_i k$ means that world j is as ‘close’ to world i as world k is.³

The worlds where a sentence $\phi \circ \psi$ is true, are

$$\bigcup_i \{j \in I : j \text{ is a } \psi\text{-world, and } j \text{ is } \leq_i \text{ minimal}\},$$

where i ranges over all ϕ -worlds.

The meaning of a counterfactual, such as ‘*If kangaroos had no tails, they would topple over*’, is that in any possible world in which kangaroos have no tails, and which resembles our own as much as possible otherwise, kangaroos would topple over (Lewis, 1973). Thus the worlds where a counterfactual $\phi > \psi$ is true are the worlds were the closest worlds in which ϕ is true also have ψ as true. In the notation used above, the worlds were $\phi > \psi$ is true are

$$\{i \in I : \text{the } \leq_i \text{ minimal } \phi\text{-worlds are also } \psi\text{-worlds}\}$$

A *model* consists of a set I of worlds, a total pre-order \leq_i over I for each $i \in I$, and a *valuation* that assigns a subset J of I to each propositional letter p . Then J is the set of p -worlds. The meaning of the Boolean connectives is the usual one, e.g. the $\phi \wedge \psi$ -worlds are the worlds that are both ϕ -worlds and ψ -worlds.

We should now be ready for a formal description of the semantic apparatus. A $\mathcal{L}_{>,o}$ -order model \mathcal{I} is a quadruple $\langle I, \leq, R, \Box \rangle$, where I is a non-empty set of ‘worlds’ i, j, k, \dots , and \leq is a function that assigns a total pre-order \leq_i over I to each member i of I , where R is a binary reflexive relation over I , and where \Box is a function that assigns a subset $[\phi]$ of I to each sentence ϕ . The relation \leq_i is a *comparative similarity ordering of worlds w.r.t. world i*. The relation R is an

³A total pre-order is a binary relation that is reflexive, transitive, and connected. Note that antisymmetry is not required. Thus it can very well be the case that j is as ‘close’ to i as k is, and that k is as ‘close’ to i as j is, without j and k being the same world.

Some concrete orders between possible worlds that have been proposed in the literature are reviewed in (Katsuno and Mendelzon, 1989).

accessibility relation. Let us introduce the notational shorthand R_i to stand for the set $\{j \in I : iR_j\}$, for each $i \in I$. Then $j \in R_i$ means that j is *accessible* from i . The function \Box is called a *valuation*. In addition, conditions (O1) – (O3), and (V1) – (V9) below have to be fulfilled.

(O1) *Centering.* If $j \leq_i i$ then $j = i$.

(O2) *Priority.* If $j \in R_i$ and $k \notin R_i$, then $k \not\leq_i j$.

(O3) *Limit assumption.* For each sentence ϕ , the set $\min_{\leq_i}([\phi] \cap R_i)$ is non-empty, whenever $[\phi] \cap R_i$ is non-empty.⁴

Condition (O1) states that no world is as close to a world i as i itself. Condition (O2) states that accessible worlds are closer than inaccessible ones. Condition (O3) guarantees that no set $[\phi] \cap R_i$ contains an infinite descending chain $j_1 >_i j_2 >_i j_3 >_i \dots$, where $j >_i k$, denotes the fact $k \leq_i j$, and $j \not\leq_i k$. A discussion of condition (O3) can be found in (Lewis, 1973), and (Nute, 1984).

The conditions for the valuation function are listed below. The conditions (V1) – (V7) state the usual Boolean semantics for the truth-functional connectives. Condition (V8) says that i is a $\phi > \psi$ -world iff all closest accessible ϕ -worlds also are ψ -worlds. Consequently, a counterfactual $\phi > \psi$ can be *vacuously* true at a world i . This happens when $[\phi] \cap R_i = \emptyset$. According to condition (V9), the result of updating ϕ with ψ consists of the closest accessible ψ -worlds, from the viewpoint of each ϕ -world separately. Thus it can happen that in a particular ϕ -world i , there are no accessible ψ -worlds. If this would be true in all ϕ -worlds i , the result of the update would be empty. Vacuous truth of a counterfactual and empty results of updates are however prevented through a condition called universality: all worlds are accessible from all other worlds (condition (R2) below).

(V1) $[\perp] = \emptyset$.

(V2) $[\top] = I$.

(V3) $[\neg\phi] = I \setminus [\phi]$.

(V4) $[\phi \wedge \psi] = [\phi] \cap [\psi]$.

(V5) $[\phi \vee \psi] = [\phi] \cup [\psi]$.

(V6) $[\phi \rightarrow \psi] = (I \setminus [\phi]) \cup [\psi]$.

(V7) $[\phi \leftrightarrow \psi] = ([\phi] \cap [\psi]) \cup ((I \setminus [\phi]) \cap (I \setminus [\psi]))$.

(V8) $[\phi > \psi] = \{i \in I : \min_{\leq_i}([\phi] \cap R_i) \subseteq [\psi]\}$.

(V9) $[\phi \circ \psi] = \bigcup_{i \in [\phi]} \min_{\leq_i}([\psi] \cap R_i)$.

A sentence ϕ is *valid* in a model $\mathcal{I} = \langle I, \leq, R, \Box \rangle$, if $[\phi] = I$. A set Σ of sentences is valid in a model \mathcal{I} if all its members are valid in the model \mathcal{I} .

⁴For $J \subseteq I$ and $i \in I$, $\min_{\leq_i}(J)$ denotes the set $\{j \in J : \text{if } k \in J \text{ and } k \leq_i j, \text{ then } j \leq_i k\}$.

Although a quadruple $\langle I, \leq, R, \Box \rangle$ that satisfies conditions (O1) – (O3) and (V1) – (V9) above counts as an order model, there are two further conditions that can be imposed on such a quadruple.

(R1) *Local uniformity.*⁵ If $j \in R_i$, then $R_j = R_i$.

(R2) *Universality.* $R_i = I$, for all $i \in I$.

An order model that satisfies (R1) is called a *locally uniform* order model, and if (R2) is satisfied, it is called a *universal* order model. The condition of interest is (R2), universality. Local uniformity is introduced primarily for technical reasons. It will turn out that the class of all locally uniform order models and the class of all universal order models validate the same set of sentences (Cf. modal logic S5).

Lemma 2.1 *Let ϕ be a $\mathcal{L}_{>,o}$ -sentence. Then ϕ is valid in all locally uniform order models if and only if ϕ is valid in all universal order models.*

In light of this lemma, the structure of universal order models can be simplified. We could thus say that a universal order model \mathcal{I} is a triple $\langle I, \leq, \Box \rangle$, drop condition (O2), and replace (V8) and (V9) with the following two conditions:

(V8') $[\phi > \psi] = \{i \in I : \min_{\leq_i}([\phi]) \subseteq [\psi]\}$.

(V9') $[\phi \circ \psi] = \bigcup_{i \in [\phi]} \min_{\leq_i}([\psi])$.

Axiomatization

Our main interest is a sound and complete axiomatization of the set of sentences that are valid in all universal order models. By Lemma 2.1 it is enough to consider the set of sentences that are valid in all locally uniform order models. Our axiom system consists of the following axioms and derivation rules.

(A1) All truth-functional axioms.

(A2) $\phi > \phi$.

(A3) $(\phi > \neg\phi) \rightarrow (\psi > \neg\phi)$.

(A4) $(\phi > \neg\psi) \vee (((\phi \wedge \psi) > \chi) \leftrightarrow (\phi > (\psi \rightarrow \chi)))$.

(A5) $((\phi \wedge \psi) > \chi) \leftrightarrow ((\psi \wedge \phi) > \chi)$.

(A6) $(\phi > \psi) \rightarrow (\phi \rightarrow \psi)$.

(A7) $(\phi \wedge \psi) \rightarrow (\phi > \psi)$.

(A8) $(\phi > \perp) \rightarrow (\neg(\phi > \perp) > \perp)$.

(A9) $\neg(\phi > \perp) \rightarrow ((\phi > \perp) > \perp)$.

The rules are the following:

(MP) *Modus Ponens.* $(\phi, \phi \rightarrow \psi) \mapsto \psi$.

⁵An equivalent definition would be to require that R is transitive and euclidean.

(CR) *Counterfactual rule.* For any $n \geq 1$,

$$\begin{aligned} ((\chi_1 \wedge \dots \wedge \chi_n) \rightarrow \psi) &\leftrightarrow \\ ((\phi > \chi_1 \wedge \dots \wedge \phi > \chi_n) \rightarrow (\phi > \psi)) \end{aligned}$$

(RR) *Ramsey's rules.* $(\chi \rightarrow (\phi > \psi)) \leftrightarrow$

$$\begin{aligned} ((\chi \circ \phi) \rightarrow \psi), \\ \text{and } ((\chi \circ \phi) \rightarrow \psi) \leftrightarrow (\chi \rightarrow (\phi > \psi)). \end{aligned}$$

Note that if we omit (RR), we have an axiomatization of Lewis' logic VCU for counterfactuals. If we in addition omit (A8) and (A9) we have an axiomatization of Lewis' VC. We therefore name our logic **VCU²**, where the second U stands for updates.

The logic **VCU²** is obtained by closing the set of all axioms by the rules. Members in this set are called $\mathcal{L}_{>,o}$ -theorems, or simply theorems. If a sentence ϕ is a theorem, we will occasionally write $\vdash \phi$.

Soundness and Completeness

It turns out that the logic **VCU²** is characterized by the class of all locally uniform order models, and therefore also by the class of all universal order models.

Theorem 2.1 *Let ϕ be a $\mathcal{L}_{>,o}$ -sentence. Then ϕ is a $\mathcal{L}_{>,o}$ -theorem, if and only if ϕ is valid in all universal order models.*

Some Validities

The next theorem lists some of the properties of our \circ -connective.

Theorem 2.2 *The following $\mathcal{L}_{>,o}$ -sentences are $\mathcal{L}_{>,o}$ -theorems.*

- (i). $(\phi \circ \psi) \rightarrow \psi$.
- (ii). $(\phi \wedge \psi) \rightarrow (\phi \circ \psi)$.
- (iii). $(\phi \circ \phi) \leftrightarrow \phi$.
- (iv). $((\phi \circ \psi) \circ \psi) \leftrightarrow (\phi \circ \psi)$.
- (v). $(T \circ \phi) \leftrightarrow \phi$.
- (vi). $(\phi \circ T) \leftrightarrow \phi$.
- (vii). $(\perp \circ \phi) \leftrightarrow \perp$.
- (viii). $(\phi \circ \perp) \leftrightarrow \perp$.
- (ix). $((\phi \vee \psi) \circ \chi) \leftrightarrow ((\phi \circ \chi) \vee (\psi \circ \chi))$.
- (x). $((\phi \wedge \psi) \circ \chi) \rightarrow ((\phi \circ \chi) \wedge (\psi \circ \chi))$.
- (xi). $(\phi \circ (\psi \vee \chi)) \rightarrow ((\phi \circ \psi) \vee (\phi \circ \chi))$.
- (xii). $((\phi \circ \psi) \wedge (\phi \circ \chi)) \rightarrow (\phi \circ (\psi \wedge \chi))$.

Monotonicity vs. Non-Monotonicity

Our logic is *monotonic* in the sense that if $\vdash \phi \rightarrow \psi$, then $\vdash (\phi \circ \chi) \rightarrow (\psi \circ \chi)$. Our logic is *non-monotonic* in the sense that if $\vdash \phi \rightarrow \psi$, then it does not necessarily follow that $\vdash (\phi \circ \chi) \rightarrow \psi$.

Decidability

Using filtration techniques, we can show that our logic has the *small model* property, and that it thus is a decidable logic.

Theorem 2.3 *There is an algorithm that, given a sentence ϕ as input, returns 'yes', if ϕ is a $\mathcal{L}_{>,o}$ -theorem, and 'no', otherwise.*

The computational complexity of this decision problem will be dealt with in a forthcoming work. Since **VCU²** includes propositional calculus, deciding theoremhood is at least as hard as in the pure propositional case. As a special case, if there are no occurrences of $>$ or \circ in the sentences ϕ and ψ , then the complexity of checking whether the 'flat' sentences of the forms $\phi > \psi$, and $\phi \circ \psi$ are theorems, is the same as for propositional calculus.

Theorem 2.4 *Let ϕ and ψ be \mathcal{L} -sentences (propositional sentences). Then the problem of deciding whether the sentence $\phi \circ \psi$ is a $\mathcal{L}_{>,o}$ -theorem, and whether the sentence $\phi > \psi$ is a $\mathcal{L}_{>,o}$ -theorem, are both in co-NP.*

3 ON UPDATES AND REVISIONS

Postulations for Change Operators

As already briefly discussed, there are (at least) two fundamentally different ways of changing a theory. The one interpretation of some new piece of knowledge is that the world has remained the same, it is only our knowledge about it that is becoming more accurate. This class of changes is called *revisions*. The other interpretation of a new piece of knowledge is that the world has undergone a change, and that we are to adapt our knowledge to the new situation. Such adaptions are called *updates*.

To highlight the distinction between the two changes, we borrow an example from Biskup (1983), who studies the problem of updating relational databases containing 'null values': Suppose that a database contains information about parts in the stock of some enterprise, and that the current state of the database is 'there are balls of an unknown color in the stock'. The change request is to 'insert' the fact that the stock now contains white balls. There are two ways in which the change request can be interpreted. If, for instance, we have inspected the stock and determined that the

existing balls are white, then the change should be a revision, and the resulting database should represent the fact that the stock contains only white balls. If, on the other hand, we have received a new shipping of white balls, the change request is obviously an update. Now the resulting database should represent the facts that there are white balls in the stock, and also balls of an unknown color, possibly white, possibly some other color. For other examples of revisions vs. updates, see (Katsuno & Mendelzon, 1991) and (Winslett, 1988).

Another fundamental distinction can be seen in the way different works model theory change. As we shall see, a change operator can either be a meta-level mapping, or an object language connective.

Gärdenfors and his colleagues have set forth a set of rationality postulates that any reasonable *revision* operator should satisfy (Alchourrón, Gärdenfors & Makinson, 1985). This approach studies revisions as mappings between abstract structures called ‘belief sets’ (essentially deductively closed theories). In particular, belief sets can be infinite. The postulates are then conditions on the mappings, and as such, formalizations of our rationality.

In the framework described in this paper the formalization of rationality appears on the level of models: there is a set of possible worlds, and a comparative similarity ordering associated with each possible world. This approach is in the spirit of Kripke models for modal logic. The update is then a connective in the object language, and the interpretation of this connective is in terms of our formalized intuition of possible worlds and similarities. In this setting, we obviously restrict ourselves to updating finite (or finitely axiomatizable) theories. We do not however regard it as a serious restriction, since any knowledgebase that is to be used in practice is inherently finite.

The two formalizations of change operators are evidently intertwined. Gärdenfors (1978) shows how to construct a system of belief sets from a $\mathcal{L}_<$ -model of a certain kind. Below we shall show that the conditions on a change operator become theorems and metatheorems in the logic having change as a connective.

Then to the rationality postulates themselves. In order to avoid introducing any new formalisms and notations, we shall use the formulation of Katsuno and Mendelzon (1989), who treat a belief set as a sentence ϕ in propositional calculus. The update operator \circ is then a mapping from pairs of propositional sentences to propositional sentences. That is, $\phi \circ \psi$ should be seen as the denotation of some propositional sentence, and the various postulates as required properties of this sentence, i.e. of the mapping \circ .

Some additional notation is however unavoidable at this point. Let S be a set of sentences containing at least all truth-functional tautologies. Then $\vdash_S \phi$ de-

notes the fact that $\phi \in S$. A sentence ϕ is said to be *consistent* in S if and only if it is not the case that $\neg\phi \in S$. The sentence ϕ is *complete* in S if and only if, for all sentences ψ in the language for S we have either $\phi \rightarrow \psi \in S$, or $\phi \rightarrow (\neg\psi) \in S$. Note that consistent (unqualified) means consistent in VCU^2 , and that $\vdash \phi$ means the same as $\vdash_{\text{VCU}^2} \phi$.

Then, let P be the set of all propositional sentences that are truth-functional tautologies. The postulates by Gärdenfors *et al* (Alchourrón, Gärdenfors & Makinson, 1985) for the mapping \circ as a *revision* function are then the following:

- (R1) $\vdash_P \phi \circ \psi \rightarrow \psi$
- (R2) If $\phi \wedge \psi$ is consistent in P , then $\vdash_P (\phi \circ \psi) \leftrightarrow (\phi \wedge \psi)$.
- (R3) If ψ is consistent in P , then $\phi \circ \psi$ is also consistent in P .
- (R4) If $\vdash_P \phi \leftrightarrow \psi$ and $\vdash_P \chi \leftrightarrow \nu$, then $\vdash_P (\phi \circ \chi) \leftrightarrow (\psi \circ \nu)$.
- (R5) $\vdash_P ((\phi \circ \psi) \wedge \chi) \rightarrow (\phi \circ (\psi \wedge \chi))$.
- (R6) If $(\phi \circ \psi) \wedge \chi$ is consistent in P , then $\vdash_P (\phi \circ (\psi \wedge \chi)) \rightarrow ((\phi \circ \psi) \wedge \chi)$.

For instance, the rationale behind (R1) is that the new sentence ψ should be true in the new knowledgebase. Postulate (R4) assures that the result of the revision is independent of the syntax of the knowledgebase ϕ and the new sentence ψ . For an intuitive account of the rest of the postulates, see (Alchourrón, Gärdenfors & Makinson, 1985).

The crucial postulate is (R2). It is precisely this postulate that is incompatible with the Ramsey Rule, and which is the major distinction between revisions and updates. If our knowledge of the real world is represented by a sentence ϕ and if we gain some additional knowledge ψ , it is of course quite plausible that our new knowledge should be represented by the sentence $\phi \wedge \psi$, in case $\phi \wedge \psi$ is consistent. But the situation for updates is different. Consider the following example from the domain of artificial intelligence (Katsuno & Mendelzon 1990) (cf. also Winslett, 1988, 1989): A room has two objects in it, a book and a magazine. Suppose p_1 means that the book is on the floor, and p_2 means that the magazine is on the floor. Let the knowledgebase be $(p_1 \vee p_2) \wedge \neg(p_1 \wedge p_2)$, i.e. either the book or the magazine is on the floor, but not both. Now we order a robot to put the book on the floor, that is, our new piece of knowledge is p_1 . If this change is to be taken as a revision, then we find that since the knowledgebase is consistent with p_1 , our new knowledgebase will be equivalent to $p_1 \wedge \neg p_2$, i.e. the book is on the floor and the magazine is not.

But the above change is inadequate. After the robot moves the book to the floor, all we know is that the

book is on the floor; why should we conclude that the magazine is not on the floor? Thus an update would have been more appropriate in this situation.

In determining a set of appropriate postulates for an update function, we have seen that (R2) cannot be used. The set of postulates for the mapping \circ as an update functions are then (R1), (R4), (R5) and (U1) – (U5) below (Katsuno & Mendelzon 1991):

- (U1) If $\vdash_P \phi \rightarrow \psi$, then $\vdash_P (\phi \circ \psi) \leftrightarrow \phi$.
- (U2) If both ϕ and ψ are consistent in P , then $\phi \circ \psi$ is also consistent in P .
- (U3) If $\vdash_P (\phi \circ \psi) \rightarrow \chi$ and $\vdash_P (\phi \circ \chi) \rightarrow \psi$, then $\vdash_P (\phi \circ \psi) \leftrightarrow (\phi \circ \chi)$.
- (U4) If ϕ is complete in P , then $\vdash_P ((\phi \circ \psi) \wedge (\phi \circ \chi)) \rightarrow (\phi \circ (\psi \vee \chi))$.
- (U5) $\vdash_P ((\phi \vee \psi) \circ \chi) \leftrightarrow ((\phi \circ \chi) \vee (\psi \circ \chi))$.

An intuitive account of these postulates is given by Katsuno and Mendelzon (1991). They also explain the model theoretic differences between updates and revisions, and give a model-theoretic characterization of the update postulates in terms of partial pre-orders on possible worlds. Since a total pre-order also is a partial one, our models satisfy the characterization. We are currently axiomatizing a logic of counterfactuals and updates based on partial pre-orders.⁶

It turns out that the \circ -connective of our logic VCU^2 satisfies the update postulates, in the sense that postulates are theorems or metatheorems in VCU^2 .

Theorem 3.1 *Let ϕ, ψ, χ , and ν be \mathcal{L} -sentences. Then*

- (R1). $\vdash \phi \circ \psi \rightarrow \psi$
- (R4). *If $\vdash \phi \leftrightarrow \psi$ and $\vdash \chi \leftrightarrow \nu$, then $\vdash (\phi \circ \chi) \leftrightarrow (\psi \circ \nu)$.*
- (R5). $\vdash ((\phi \circ \psi) \wedge \chi) \rightarrow (\phi \circ (\psi \wedge \chi))$.
- (U1). *If $\vdash \phi \rightarrow \psi$, then $\vdash (\phi \circ \psi) \leftrightarrow \phi$.*
- (U2). *If both ϕ and ψ are consistent, then $\phi \circ \psi$ is also consistent.*

⁶The natural question at this point is whether there exists a finite axiomatization of a logic with updates but no counterfactuals. Indeed, it could be possible to replace the axioms (A2) – (A7) with axioms that only includes the \circ -connective. But in order to verify (U2), we need the effect of axioms (A8) and (A9). It seems that this effect is not achievable without at least a modal operator of necessity. Therefore, we might as well use the $>$ -connective, since it is more natural in the present context. Note that the sentence $(\neg\phi) > \perp$ has the same meaning as the sentence $\Box\phi$ in modal logic S5 (modal logic T, in absence of (R1) and (R2) and (A8) – (A9)).

(U3). *If $\vdash (\phi \circ \psi) \rightarrow \chi$ and $\vdash (\phi \circ \chi) \rightarrow \psi$, then $\vdash (\phi \circ \psi) \leftrightarrow (\phi \circ \chi)$.*

(U4)⁷. *If ϕ is complete in P , then $\vdash ((\phi \circ \psi) \wedge (\phi \circ \chi)) \rightarrow (\phi \circ (\psi \vee \chi))$.*

(U5). $\vdash ((\phi \vee \psi) \circ \chi) \leftrightarrow ((\phi \circ \chi) \vee (\psi \circ \chi))$.

A Non-Triviality Result

Gärdenfors (1986, 1988) has shown that any logic with the Ramsey Rule and with a change operator satisfying postulates (R1), (R2) and (R3) is *trivial* in a certain sense. Gärdenfors shows his theorem in the context of belief sets, but it can be rephrased and reproved in the current formalism. In the following we assume that all logics include the truth functional tautologies.

A logic L is said to be *non-trivial* if there are at least four sentences ϕ, ψ, χ , and μ in the language for L , such that the sentences $\psi \wedge \chi, \psi \wedge \mu$, and $\chi \wedge \mu$ are inconsistent in L , and the sentences $\phi \wedge \psi, \phi \wedge \chi$, and $\phi \wedge \mu$ are consistent in L . Otherwise the logic L is *trivial*.

Gärdenfors' Triviality Theorem 3.2 *Let L be a logic with \circ and $>$ connectives, such that L satisfies Ramsey's Rules (RR) and postulates (R1), (R2), and (R3). Then L is trivial.*

The proof of Gärdenfors' Triviality Theorem does not apply to VCU^2 , since postulate (R2) is not satisfied. We can also explicitly show that our logic is non-trivial, as well as consistent.

Theorem 3.3 *The logic VCU^2 is consistent and non-trivial.*

4 CONCLUSIONS

Consider a counterfactual ‘If A , then B ’. The Gärdenfors Triviality Theorem shows that the antecedent cannot be read as ‘If I knew, possibly contrary to my previous beliefs, that A were the case’. On the other hand, our results show that the antecedent consistently can be read as ‘If the world was such that A were the case’. This reading seems to be the intended one. It does not of course exclude the existence of other kind of counterfactuals.

From the point of view of theory change, the logic VCU^2 provides a semantics for updating knowledge-bases, with the obvious advantage of decidability. The question of how to specify the orders on possible worlds, pertaining to some particular application — an analogue to the *frame axioms* — requires further studies. One possible avenue would be to assign priorities to the sentences in the knowledgebase, as in

⁷In this case we assume that the language is finitary.

(Gärdenfors & Makinson, 1988) and (Fagin, Ullman & Vardi, 1983).

Acknowledgements

The connection between counterfactuals and updates was first recognized in (Katsuno & Mendelzon, 1991). Without the numerous discussions the present author had with Alberto Mendelzon, this paper would not have been possible. Thanks also go to Craig Boutilier and Hirofumi Katsuno for their comments on an earlier draft.

References

- S. Abiteboul & G. Grahne (1985). Update semantics for incomplete databases. In *Proceedings of the 11th International Conference on Very Large Databases*, pp. 1-12.
- C. E. Alchourrón, P. Gärdenfors & D. Makinson (1985). On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510-530.
- H. L. Arló Costa (1989). Conditionals and monotonic belief revisions: the success postulate. In *VI Simposio Latinoamericano de Lógica Matemática*.
- J. Biskup (1983). A foundation of Codd's relational maybe-operations. *ACM Trans. on Database Systems*, 8:608-636.
- A. J. Bonner (1988). Hypothetical Datalog. In *Proceedings of the Second International Conference on Database Theory*, pp. 144-160.
- C. B. Cross (1989). Belief revision, nonmonotonic reasoning, and the Ramsey test. In *Defeasible Reasoning and Knowledge Representation*. H. E. Kyburg et al, editors. Kluwer, Boston.
- R. Fagin, J. D. Ullman & M. Y. Vardi (1983). On the semantics of updates in databases. In *Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp. 352-365.
- M. L. Ginsberg (1986). Counterfactuals. *Artificial Intelligence*, 30:1.
- P. Gärdenfors (1988). *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Bradford Books, MIT Press, Cambridge, MA.
- P. Gärdenfors (1978). Conditionals and changes of belief. *Acta Philosophica Fennica*, Vol. XXX, pp. 381-404.
- P. Gärdenfors (1986). Belief revisions and the Ramsey test for conditionals. *The Philosophical Review*, XCV, No 1:81-93.
- P. Gärdenfors & D. Makinson (1988). Revisions of knowledge systems using epistemic entrenchment. In *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pp. 83-95.
- J. Hintikka (1962). *Knowledge and belief*. Cornell University Press, New York.
- H. Katsuno & A. O. Mendelzon (1989). A unified view of propositional knowledge base updates. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pp. 1413-1419.
- H. Katsuno & A. O. Mendelzon (1990). Propositional knowledge base revision and minimal change. Technical Report KRR-TR-90-3, Department of Computer Science, University of Toronto.
- H. Katsuno & A. O. Mendelzon (1991). On the difference of updating a knowledgebase and revising it. In Allen, J.A., Fikes, R., and Sandewall, E. (eds.) *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*. San Mateo, CA: Morgan Kaufman. See also: Technical Report KRR-TR-90-6, Department of Computer Science, University of Toronto.
- A. M. Keller & M. Winslett Wilkins (1985). On the use of an extended relational model to handle changing incomplete information. *IEEE Trans. on Software Engineering*, SE-11:7, pp. 620-633.
- I. Levi (1988). Iteration of conditionals and the Ramsey test. *Synthèse*, 76. pp. 49-81.
- D. K. Lewis (1973). *Counterfactuals*. Blackwell, Oxford.
- D. Makinson (1985). How to give it up: A survey of some formal aspects of the logic of theory change. *Synthèse*, 62:347-363.
- D. Makinson (1990). The Gärdenfors impossibility theorem in non-monotonic contexts. To appear in *Studia Logica*.
- D. Nute (1984). Conditional logic. In Dov M. Gabbay and Franz Guenther, editors, *Handbook of Philosophical Logic*, D. Reidel, Dordrecht.
- F. P. Ramsey (1950). General propositions and causality. In *Foundations of Mathematics and Other Logical Essays*, R. B. Braithwaite, editor. Routledge and Kegan Paul, New York, pp. 237-57.
- H. D. Rott (1986). Ifs, though and because. *Erkenntnis* 25, pp. 345-370.
- M. Winslett (1988). Reasoning about action using a possible models approach. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp. 89-93.
- M. Winslett (1989). Theory revision semantics for use in reasoning about action. Unpublished manuscript.
- M. Winslett (in press). *Updating Logical Databases*. Cambridge University Press.

Probably Approximately Optimal Derivation Strategies

Russell Greiner

Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4

Pekka Orponen*

Department of Computer Science
University of Helsinki
SF-00510 Helsinki, Finland

Abstract

An inference graph can have many “derivation strategies”, each a particular ordering of the steps involved in reducing a given query to a sequence of database retrievals. An “optimal strategy” for a given distribution of queries is a complete strategy whose “expected cost” is minimal, where the expected cost depends on the conditional probabilities that each requested retrieval succeeds, given that a member of this class of queries is posed. This paper describes the PAO algorithm that first uses a set of training examples to approximate these probability values, and then uses these estimates to produce a “probably approximately optimal” strategy — i.e., given any $\epsilon, \delta > 0$, PAO produces a strategy whose cost is within ϵ of the cost of the optimal strategy, with probability greater than $1 - \delta$. This paper also shows how to obtain these strategies in time polynomial in $1/\epsilon, 1/\delta$ and the size of the inference graph, for many important classes of graphs, including all and-or trees.

1 Introduction

General problem solving (a.k.a. deduction) is expensive. There can be an exponential number of potential “solution paths” for a given query/goal — as there can be many rules/operations that each reduce the goal to a new set of subgoals, and each of these subgoals can, itself, have many possible reductions, etc. [GN87]. It is obviously advantageous to explore these different paths in a good order, one that will lead to a solution quickly. This “best ordering” depends on the probabilities that each path will succeed; unfortunately, this information is usually unavailable. In addition, the

task of finding the optimal strategy, even given these probabilities, can be very difficult; c.f., [Gre91b]. This report presents an efficient technique for finding good approximations to these probabilities, and then using them to produce near-optimal derivation systems.

The rest of this section provides the framework for this research, and discusses how it relates to the work on satisficing search strategies, effective use of control information, and explanation-based learning. Section 2 then describes our underlying task: designing efficient algorithms that first use samples to approximate an unknown distribution, and then use this estimate to produce good strategies, with provably high probability. The following two sections present such algorithms, for various classes of situations: Section 3 first presents a relatively simple version, PAO₀, that is sufficient for one class of knowledge bases. Section 4 then presents various modifications to this algorithm, producing systems that are more efficient and are able to handle other situations.

Framework: The objective of a derivation process (DP) is to determine whether a given query follows from a given knowledge base (KB). This work focuses on “backward chaining derivation processes”, atomic literal queries and non-recursive “definite clause KBs”.¹ Here, an atomic formula σ (e.g., the initial query) follows from the KB if a matching fact is in the KB (i.e., if the “database retrieval” for σ succeeds) or if there is a rule whose conclusion matches σ and whose antecedents all follow from the KB. Our DP can, therefore, backward-chain from the query through the rules, to reach a *satisficing set* of retrievals — i.e., the query follows whenever each of these retrievals succeeds. (E.g., consider the trivial knowledge base KB_C , whose rules appear in Figure 1. Here, the singleton $\{D_c\}$ set of retrievals (corresponding to the $Cheap(C1)$ literal) forms a *satisficing set* for the $BuyCar(C1)$ query, based on the R_{bc} rule.)

*This research was performed while this author was visiting the Department of Computer Science at the University of Toronto.

¹That is, every clause in the KB’s CNF form contains exactly one positive literal. The atomic clauses are called “facts”, and the others, “rules”.

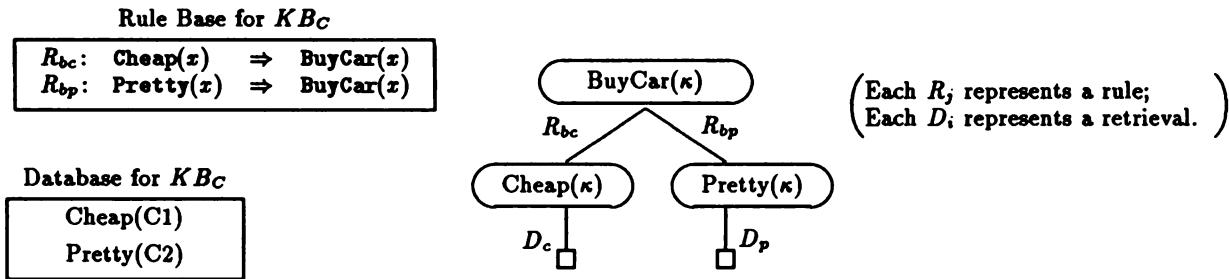


Figure 1: The KB_C knowledge base and its associated inference graph, G_C

In general, the DP can use the KB's rules to reduce the query to various satisfying sets of retrievals. A satisfying set of retrievals is a *solution* if each member of that set succeeds — i.e., if each of these literals is present in the KB. Notice that different satisfying sets will be solutions for different queries. (E.g., as $\{\text{Cheap}(C1), \text{Pretty}(C2)\}$ are all of KB_C 's facts, $\{D_c\}$ represents a solution for the $\text{BuyCar}(C1)$ query, but not for either $\text{BuyCar}(C2)$ or $\text{BuyCar}(C19)$.)

It is convenient to arrange these rules into an “inference graph”, where each node corresponds to an atomic formula that appears in some rule, and each arc, to a rule or a database retrieval; the tree in Figure 1 is an example. Derivation can then be viewed as a search through this graph, trying to find some solution.

We can therefore reduce our task of building an efficient derivation process to a graph theoretic task, of finding a good “strategy”, where a strategy is a fixed ordering on these arcs, specifying when to use each rule and when to perform each retrieval. (For example, given the query $\text{BuyCar}(C3)$, the sequence $\Theta_1 = \langle R_{bc} D_c R_{bp} D_p \rangle$ represents the strategy that first follows the R_{bc} rule down to the $\text{Cheap}(C3)$ database retrieval, D_c . If that retrieval succeeds, the derivation process returns “success” and terminates. If not, it then pursues the strategy’s other sub-path: following the R_{bp} rule down to the $\text{Pretty}(C3)$ database retrieval, D_p . The derivation process then returns either success or failure, based on the success of this second retrieval.) An alternative to this strategy, $\Theta_2 = \langle R_{bp} D_p R_{bc} D_c \rangle$ first follows R_{bp} to D_p and then, if needed, R_{bc} to D_c . Which of the two strategies is better?

A “good” strategy is one that performs well in practice. To quantify this, we assume there is some distribution of queries that our derivation system will be asked to solve. We can then define a strategy’s *expected cost* as the average cost² required to find one solution for a query (or to determine that no solution exists) averaged over the distribution of anticipated

queries, Q . This value depends on the computational cost of each reduction and retrieval, and on the *a priori* likelihoods that each particular retrieval will succeed — i.e., the conditional probability that a specific requested proposition will be in the KB, given that the query is taken from Q . (Continuing with the same KB_C , assume that each reduction and retrieval costs 1 unit, and imagine we knew that 60% of the queries would be $\text{BuyCar}(C1)$, 25% would be $\text{BuyCar}(C2)$, and 15%, $\text{BuyCar}(C3)$. Here, the D_c retrieval will succeed 60% of the time, and D_p , 25%. We can now compute Θ_1 ’s expected cost, $E[\Theta_1]$: it is the cost of the initial $\langle R_{bc} D_c \rangle$ subpath, plus the probability that this path will fail ($1 - 0.6 = 0.4$) times the cost of $\langle R_{bp} D_p \rangle$; hence, $E[\Theta_1] = (1 + 1) + 0.4(1 + 1) = 2.8$.) In general, a strategy’s expected cost is the weighted sum of the costs of each path to its set of retrievals, weighted by the probability that we will need to pursue this path (i.e., that none of the prior paths succeeded).³

We can now state our objective: to find a graph traversal strategy that is complete (i.e., is guaranteed to find a solution if there is one) and has the minimal expected cost of all such complete strategies [Gre91a]. We consider such strategies to be *optimal*.

There are several known algorithms for computing an optimal strategy for various classes of KBs (or for the related, more abstract task of finding the optimal *satisficing search strategy* for various general search structures [OG90]). Most, including [Gar73, Bar84, SK75, Nat86, Gre91a], assume that the success probabilities are given initially. Unfortunately, these probabilities are, in general, not known initially (e.g., we usually do not know that 60% of the queries will be $\text{BuyCar}(C1)$, etc.). [Smi89] presents one way of approximating their values, based on the (questionable) assumption that these probabilities are correlated with the distribution of facts in the database. For example, assume that KB'_C includes the same rule base as KB_C together with 1,000 facts of the form $\text{Cheap}(\kappa)$ (each using a different constant κ) and 10 facts of the form $\text{Pretty}(\kappa)$. Given any goal of the form $\text{BuyCar}(\kappa)$

²While we usually think of this “cost” as time, it could also be number of external file reads, or any other positive measure.

³It is actually more complicated, as we need only consider the cost of the *additional steps*, to go from some already-visited nodes down to the retrievals. See [Gre91a].

for some constant κ — e.g., BuyCar(C19) — that system assumes that we are 100 times more likely to find the corresponding **Cheap** fact (here “**Cheap(C19)**”) than the **Pretty** fact (“**Pretty(C19)**”). This means [Smi89]’s algorithm would claim that Θ_1 is the optimal strategy (i.e., it is better than the other legal candidate, Θ_2).

This, of course, need not be true; there is no reason to assume the distribution of the user’s queries will be correlated with the distribution of facts in the KB. The user may, for example, only ask questions whose answers just happen to come from the **Pretty** facts. (For example, imagine that all of the queries dealt with RollsRoyces — i.e., were of the form $\text{BuyCar}(\kappa)$, where κ referred to a not-cheap Rolls Royce.) What we really need is the *conditional probability that a given retrieval will succeed, given that a query from a particular class is asked* — i.e., $\Pr[\text{Cheap}(\kappa) \text{ retrieval succeeds} \mid \text{query } \text{BuyCar}(\kappa) \text{ is asked}]$. That is, imagine the user asks a total of K $\text{BuyCar}(\kappa)$ queries over the lifetime of the system (e.g., BuyCar(C1) , BuyCar(C19) , BuyCar(C2) , ...); and of these, the related $\text{Cheap}(\kappa)$ retrieval (if requested) would succeed a total of $\#c(K)$ times. Then the value we need, the real value of $\Pr[\text{Cheap}(\kappa) \text{ succeeds} \mid \text{BuyCar}(\kappa) \text{ asked}]$, is the ratio of these numbers — $\frac{\#c(K)}{K}$.

While there is no empirical way of finding such probabilities exactly (short of running all K trials), there is a meaningful way of estimating their values, based on the observation that the past can be a good predictor of the future. This basic observation is the key insight underlying “explanation-based learning” systems [MKKC86, DM86, MCK⁺89]. Each of these systems analyzes the solutions found to certain previous problems, and many use this information to suggest new strategies. Most of these systems, however, use only a single training example. Empirical studies have confirmed that the resulting strategies are not always good [Min88].

By contrast, this research uses a *set* of examples, and can guarantee that the resulting strategy will usually be arbitrarily close to optimal. In particular, this paper presents an efficient process that determines the number of training examples needed to approximate these probability values with the property that, with high probability, these estimates will be good enough to produce an approximately optimal derivation strategy. It assumes only that there is a set of queries that our DP will encounter, and that these queries are selected at random, according to some arbitrary but stationary distribution. (Notice this distribution will depend on the particular task our DP is addressing; *n.b.*, we do not assume that it is a uniform distribution over all possible problems [Gol79], nor that it will necessarily correspond to any particular collection of “benchmark challenge problems” [Kel87]. Hence,

we are using the same weak “distribution-free assumption” that underlies the current work in PAC-learning [Val84].) We can translate this query distribution into a set of probability values that specify the chance that each retrieval will succeed. We assume, also, that these success probabilities (of the different retrievals) are independent of one another.

2 The PAO Task

This section defines our task in very abstract terms; the next two sections present concrete algorithms that achieve this task, in various situations.

As mentioned above, a derivation process, DP , is a program that answers queries based on the information in a knowledge base. The cost it requires to produce that answer will depend on the particular *strategy* it uses. We assume there is a class of strategies associated with any given knowledge base (e.g., Θ_1 and Θ_2 are the two strategies associated with KB_C) and define DP_Θ to be the process that uses the strategy Θ .

Each strategy will perform various “probabilistic experiments” (e.g., attempted database retrievals), which each succeed for certain queries, and fail for others. (E.g., the two experiments associated with KB_C are D_c and D_p , corresponding to the $\text{Cheap}(\kappa)$ and $\text{Pretty}(\kappa)$ attempted database retrievals; notice D_c succeeds when the query is BuyCar(C1) , but fails when the query is BuyCar(C2) .) We can, in general, identify each query with the subset of the experiments that succeed in that context: e.g., using the \mathcal{L} function that maps each query to the associated set of successful probabilistic experiments, $\mathcal{L}(\text{BuyCar(C1)}) = \{D_c\}$ and $\mathcal{L}(\text{BuyCar(C19)}) = \{D_p\}$.

We define $e[\Theta, q]$ to be the cost for DP_Θ to answer the query q . (Notice this depends only on which experiments succeed; i.e., on the value of $\mathcal{L}(q)$.)

We assume that the queries are drawn from a set Q at random, according to some stationary distribution. This means we can consider the set of $e[\Theta, q_i]$ to be independent, identically-distributed random variables, and then define $E[\Theta]$ to be their common expected value; i.e.,

$$E[\Theta] = \text{average}_{q \in Q} e[\Theta, q]$$

As we further assume that the experiments are independent, we can view $E[\Theta]$ as a function of their success probabilities; to make this explicit, we will write $E_P[\Theta]$, where P is the vector of the probability values for the various experiments. (E.g., $P = (p_c, p_p)$ where $p_c = \Pr[\text{Cheap}(\kappa) \text{ succeeds} \mid \text{BuyCar}(\kappa) \text{ asked}]$ and $p_p = \Pr[\text{Pretty}(\kappa) \text{ succeeds} \mid \text{BuyCar}(\kappa) \text{ asked}]$.)

We can then define the *optimal strategy* as the complete strategy whose expected cost is minimal, with

respect to the distribution (i.e., probability vector, P):

$$\Theta_{opt} \text{ is optimal} \iff \forall \Theta. E_P[\Theta_{opt}] \leq E_P[\Theta]$$

We can now define the PAO task: Each problem instance takes

- \mathcal{S} , the set of possible derivation processes, all based on a given KB
(e.g., the derivation processes D_{P_1} and D_{P_2} associated with KB_C , based on the Θ_1 and Θ_2 strategies, respectively)
- \mathcal{Q} , a (stationary) distribution of the queries
(e.g., $\{\text{BuyCar}(C1), \text{BuyCar}(C3), \text{BuyCar}(C7), \dots\}$)
- \mathcal{O} , an oracle that generates appropriate queries from \mathcal{Q} according to this fixed (but unknown) distribution,
- $\epsilon \in \mathbb{R}^+$ (i.e., a bound on the allowed excess), and
- $\delta \in (0, 1]$ (i.e., the required confidence).

For each instance, we seek a strategy $\Theta_{pao} \in \mathcal{S}$, whose expected cost is, with high probability, close to the cost of the optimal strategy. Stated more precisely, given the real distribution (here, the $P = (p_c, p_p)$ defined above), there is an optimal strategy, $\Theta_{opt} \in \mathcal{S}$, whose expected cost is $E_P[\Theta_{opt}]$. With high probability (i.e., $\geq 1 - \delta$), the cost of the result of this algorithm, $E_P[\Theta_{pao}]$, will be no more than ϵ over $E_P[\Theta_{opt}]$ — i.e.,

$$\Pr [E_P[\Theta_{pao}] \leq E_P[\Theta_{opt}] + \epsilon] \geq 1 - \delta.$$

Notes: [1] We insist that the algorithm for this task be efficient — i.e., run in time that is polynomial in $1/\epsilon$, $1/\delta$ and the “size” of knowledge base.

[2] Each PAO algorithm will be responsible for computing the \mathcal{L} function — i.e., it must use the knowledge base to determine which experiments are successful, for each observed query.

[3] The oracle can simply be the *user* of the DP system, who is going about his usual business of asking the system relevant queries. (That is, he can ask his questions during the “training mode”, while PAO is computing the pao strategy, as well as afterwards.) Of course, the overall DP+PAO system would have to return the solution found (as a side-effect, wrt PAO).

3 Simple PAO₀ Algorithm

This section presents the (relatively simple but general) PAO₀ algorithm that, with high probability, identifies approximately optimal strategies for a certain specified class of KBs (*viz.*, non-recursive definite class KBs whose inference graphs are hyper-trees; see below). After Subsection 3.1 presents the necessary definitions, Subsection 3.2 describes the algorithm itself. Subsection 3.3 sketches the proofs that PAO₀ works correctly and is efficient. The next section presents several extensions to this basic algorithm, designed to provide answers more efficiently and to handle other situations.

3.1 Definitions

The definitions below define the particular \mathcal{S} used, and constrain \mathcal{Q} :

- We can identify any non-recursive definite clause KB with a particular inference graph $G = (N, A, S, B, c)$, where N is a set of nodes, each node corresponds to an atomic literal that must be proven towards achieving some goal; $A \subset N \times 2^N$ is the set of directed hyper-arcs, each arc corresponds either to an invocation of a rule or a database retrieval.⁴ The set $S \subset 2^N$ corresponds to the set of “success node-sets”; reaching each element of some member (i.e., reaching each $n \in s$ for some $s \in S$) means the overall derivation is successful. (For the G_C inference graph associated with the KB_C knowledge base, these sets are the singletons $\{D_c\}$ and $\{D_p\}$, as we need only find a single successful database retrieval to establish the given goal in this simple context.⁵) The set $B \subset A$ corresponds to the “blockable” arcs — each $b \in B$ is an arc that can be “blocked” (i.e., not succeed) for some query. (E.g., D_c is blockable as it does not succeed for the $\text{BuyCar}(C2)$ query.) The *cost function* $c: N \times N \mapsto \mathbb{R}^+$ gives the positive real valued cost of each part of a hyper-arc in G .⁶

PAO₀ deals only with inference graphs that are “hyper-trees”; i.e., connected directed graphs, each with a unique “root” node (having no parents), where every other node has a single parent. (See Subsection 4.2.)

As the KB_C knowledge base is “disjunctive” (i.e., it is definite clause and each of its CNF clauses includes at most one negative literal), its associated inference graph, G_C , relatively easy — in particular, each node descends to a single child node. Appendix A provides an example of a less trivial hyper-tree, based on non-disjunctive KB.

- Given any such inference hyper-tree G , we let $\mathcal{S}(G) = \{D_{P_e}\}$ represent the set of derivation processes, where each D_{P_e} uses its own strategy Θ to specify how it will search this inference graph to find an answer to a given query. In the case of simple disjunctive KBs, each strategy corresponds

⁴The graph only includes an arc corresponding to a database retrieval if the corresponding proposition *might* be in the fact set. Notice, for example, that the G_C inference graph does not include an arc corresponding to the $\text{BuyCar}(\kappa)$ retrieval, as the KB’s fact set does not include any such propositions.

⁵ S includes only the *minimal* success sets; *n.b.*, any superset of any member of S is also sufficient. Hence, the set $\{D_c, D_p\}$ is sufficient.

⁶Notice these values are not used when answering queries; they are, however, essential for parts of our PAO algorithms.

to a sequence of the elements of A ;⁷ DP_Θ follows these arcs, in this order, stopping as soon as it reaches a success node. (E.g., the DP_Θ process, based on the $\Theta_1 = \langle R_{bc}, D_c, R_{bp}, D_p \rangle$ strategy, will stop if the D_c retrieval is successful, ignoring the remaining $\langle R_{bp}, D_p \rangle$.) Notice this means that our DPs are seeking only a single solution; this type of search is called a “satisficing search” [SK75].⁸ Strategies are considerably more complicated when dealing with non-disjunctive KBs; see Appendix A.

- Each query (i.e., each member of Q) corresponds to an instantiation of the root node of G .

Hence, each DP_Θ is given a query, q , as its input. It traverses the inference graph G , beginning at the root node (corresponding to q), in the order described by Θ , trying to reach a satisfying set of success nodes (i.e., all nodes in some member of S). DP_Θ is only allowed to use the arcs that are not blocked for this query, i.e., the members of $\mathcal{L}(q)$.

In general, the expected cost of a strategy, $E[\Theta]$, can be expressed as a function of the success probabilities of the various experiments, and the values of the c cost function; see Appendix A.

3.2 Basic PAO₀ Algorithm

The PAO₀ algorithm presented in this subsection deals with satisficing searches through any KB whose inference graph is a hyper-tree. It consists of three subroutines, GET-NUMBER₀, FIND-PROBS₀ and FIND-OPTDS₀, that are called in sequence. The GET-NUMBER₀ procedure takes as input an inference graph, G , and the two “error terms”, ϵ and δ , and returns an integer

$$M = \mathcal{GN}_{AOG}(G, \epsilon, \delta) \stackrel{\text{def}}{=} \lceil 2 \left(\frac{nC}{\epsilon} \right)^2 \ln \frac{2n}{\delta} \rceil \quad (1)$$

where $C = \sum_{a \in A} c(a)$ is the sum of the costs of all of the arcs in the inference graph G , and $n = |B|$ is the total number of probabilistic arcs.

The FIND-PROBS₀ procedure then calls the \mathcal{O} oracle M times to obtain M queries from the class Q . For each of these $q_i \in Q$, it must determine the value of $\mathcal{L}(q_i)$ — i.e., which of the probabilistic arcs are blocked. (E.g., it must determine, for the BuyCar(C1) query, that the D_c retrieval succeeds but the D_p retrieval does not.) This involves searching through the

⁷Of course, not every sequence qualifies; in particular, we only consider sequences where each arc descends from a node that has been already been reached. Hence, $\langle D_p, R_{bp}, \dots \rangle$ is not a legal strategy as its first step, the D_p arc, descends from a node that has not yet been reached.

⁸Notice it applies to ground queries as well as existentially quantified queries, when the user is seeking only a single binding list for those variables.

entire inference graph. We implement it using the DP⁽¹⁾ derivation process, that finds *all* solutions to a given query; i.e., it goes down all possible paths, and attempts each probabilistic arc.⁹

After all M queries, FIND-PROBS₀ tallies how many times each probabilistic arc succeeded, and translates this number into a frequency value. (Continuing the above example, assume that GET-NUMBER₀ returns the value $M_{GC} = 30$ for the GC graph and some values of ϵ and δ . FIND-PROBS₀ would then pose $M_{GC} = 30$ queries, each of the form $\text{BuyCar}(\kappa)$ for some appropriately-distributed set of ground κ s, to the KBC knowledge base and observe that the associated $\text{Cheap}(\kappa)$ query (resp., $\text{Pretty}(\kappa)$ query) succeeded 10 times (resp., 18 times). It would then produce the frequency values of $\hat{p}_c = \frac{10}{30}$ and $\hat{p}_p = \frac{18}{30}$.)

FIND-PROBS₀ gives these approximate probability values and the inference graph to the FIND-OPTDS₀ procedure, that produces the ϵ, δ -probably-approximately-optimal derivation strategy, Θ_{pao} . FIND-OPTDS₀ is simply the $\text{T}_{AO}\mathcal{T}(G, Q)$ algorithm (defined in [Gre91a]) that takes G , a general hyper-tree, and Q , a vector of (approximations to) the success probabilities of the retrievals, and returns the strategy that would be optimal, if these Q values were accurate; i.e., $\Theta_Q = \text{T}_{AO}\mathcal{T}(G, Q)$ satisfies $\forall \Theta \in \mathcal{S}(G) E_Q[\Theta] \geq E_Q[\Theta_Q]$. PAO₀ then returns this value, $\Theta_P = \text{T}_{AO}\mathcal{T}(G, \hat{P})$.

3.3 Proof Sketches

The critical claim underlying the PAO₀ algorithm is the formula in Equation 1, which is proven in [GO91, Theorem 1]. The basic ideas follow.

Let G be an inference graph with n probabilistic arcs; and $P = \langle p_1, \dots, p_n \rangle$ be the real probability vector — i.e., p_i is the (real) success probability of the i^{th} probabilistic arc. The real optimal strategy for the inference graph G , based on these probability values, is $\Theta = \text{T}_{AO}\mathcal{T}(G, P)$. The expected cost of this strategy (again based on the real probability vector P) is $E_P[\Theta]$. Imagine, however, that we only have an approximation to P , called \hat{P} ; the “optimal” strategy, based on this vector, is $\hat{\Theta} = \text{T}_{AO}\mathcal{T}(G, \hat{P})$. How bad is this $\hat{\Theta}$ — that is, how much more might this strategy cost, over the optimal one? This depends on how close each p_i is to the corresponding component of \hat{P} (called \hat{p}_i). The answer appears below:

Lemma 3.1 (Sensitivity Analysis of $\text{T}_{AO}\mathcal{T}(G, P)$) Let $G = \langle N, A, S, B, c \rangle$ be an inference graph with

⁹Notice this DP⁽¹⁾ differs from the DP₀s, as each DP₀ will stop as soon as it finds a solution. Hence, DP₀₁ will stop if the D_c retrieval succeeds, and will not consider the $\langle R_{bp}, D_p \rangle$ path. By contrast, DP⁽¹⁾ will always try both D_c and D_p , even if the initial D_c retrieval succeeds.

n probabilistic arcs $B = \{b_1, \dots, b_n\}$, and let $P = (p_1, \dots, p_n)$ and $\hat{P}_i = (\hat{p}_1, \dots, \hat{p}_n)$ be two n -element probability vectors whose respective i^{th} components differ by at most $\lambda_i \in \mathbb{R}$; i.e., $\forall i. |p_i - \hat{p}_i| \leq \lambda_i$. Then

$$E_P[\text{TAOT}(G, \hat{P})] \leq E_P[\text{TAOT}(G, P)] + 2 \sum_i \lambda_i \cdot C_{\neg}[b_i]$$

where $C_{\neg}[b_i]$ is the sum of the costs of the arcs in G that are not on the path from the G 's root, through b_i , to a success node.

(To illustrate the $C_{\neg}[\cdot]$ function: In the G_C graph, the value of $C_{\neg}[D_p]$ is the sum of the costs of the R_{bc} rule and the D_c retrieval; i.e., $C_{\neg}[D_p] = c(R_{bc}) + c(D_c)$. Likewise, $C_{\neg}[D_c] = c(R_{bp}) + c(D_p)$. Notice each $C_{\neg}[b_i] \leq C$, where $C = \sum_{a \in A} c(a)$ is the sum of the costs of all of the arcs in G .)

Given any ϵ , therefore, we need to sample each experiment enough times that we are confident that each \hat{p}_i is within $\lambda_i = \frac{\epsilon}{2nC}$ of the real value, p_i . To show that $\mathcal{GN}_{AOG}(G, \epsilon, \delta)$ trials of each experiment is sufficient, we need only the assertion that these experiments are independent, and the following simple form of Chernoff's bound:

Lemma 3.2 (from [Bol85, p. 12]) *If M independent experiments are performed to obtain an estimate \hat{p} for a probability p , and $\lambda \geq 0$, then*

$$\Pr[|\hat{p} - p| > \lambda] \leq 2e^{-2M\lambda^2}$$

Finally, this algorithm is efficient:

Lemma 3.3 (PAO₀'s Efficiency) *The PAO₀ algorithm runs in time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$ and $|G|$, the size of the inference graph (assuming a constant time oracle \mathcal{O} and constant time database retrieval process).*

(The proof involves the observations that GET-NUMBER₀ requires only $O(|G|)$ to compute the values of n and C ; FIND-PROBS₀ requires only $O(\frac{1}{\epsilon} \frac{1}{\delta} |G|^3)$ (i.e., $O(|G|)$) to traverse the inference graph to perform the n retrievals for each of its $M = O(\frac{1}{\epsilon} \frac{1}{\delta} |G|^2)$ trials); and FIND-OPTDS₀ is $O(|G|^2)$ [Gre91a].)

4 Extensions to the PAO Algorithm

For pedagogical reasons, the previous section presented a very simple algorithm, one that is both relatively restricted (able to handle only inference hyper-trees¹⁰) and relatively inefficient. This subsection

¹⁰Technically, we should state "KBs whose inference graphs are hyper-trees". To simplify our descriptions, we will often identify a class of KBS with their class of inference graphs.

presents several alternate PAO_i algorithms that are more efficient or can process other situations (e.g., other classes of inference graphs, etc.). ([GO91] discusses yet other extensions to the basic PAO₀ algorithm.)

4.1 Fewer Total Number of Retrievals

The FIND-PROBS₀ part of the PAO₀ system must perform $M \cdot n$ retrievals for a general and-or inference tree — i.e., it must perform all n retrievals on each of M trials. We show below two modifications that allow FIND-PROBS to perform a smaller total number of retrievals.

Use Different Formulae for Different Classes of Graphs: The \mathcal{GN}_{AOG} formula is an upper bound, that applies to any and-or inference tree.¹¹ Lower values apply to certain subclasses; for example, we need only

$$\mathcal{GN}_{2o}(G, \epsilon, \delta) \stackrel{\text{def}}{=} \lceil 2 \left(\frac{c_r}{\epsilon} \right)^2 \ln \frac{2}{\delta} \rceil$$

trials for any inference graph of the form shown in Figure 1 — i.e., involving exactly two simple (non-hyper) paths, each leading to a singleton solution set through a single probabilistic arc, each path of equal cost, called c_r (e.g., for G_C , this is the cost of each rule plus its associated retrieval; i.e., $c_r = c(R_{bc}) + c(D_c) = c(R_{bp}) + c(D_p)$).¹² Here, FIND-PROBS would have to perform only $\mathcal{GN}_{2o}(\dots) < \mathcal{GN}_{AOG}(\dots)$ trials.

The GET-NUMBER₁ algorithm¹³ can exploit such formulae. This process first identifies the given graph with the appropriate formula, \mathcal{GN}_G (where some appropriate set of formulae, $\{\mathcal{GN}_G\}$, is known in advance), then evaluates that \mathcal{GN}_G formula on the appropriate values, corresponding to various numbers associated with this graph (e.g., the number of probabilistic arcs, particular costs of the arcs, etc.).

The classification part is relatively straightforward, usually based on certain properties of the KB, such as whether it is propositional (rather than in general predicate calculus), or definite, or disjunctive, etc. We consider only simple tests that can be run in time polynomial in the size of the inference graph; notice all three of the tests mentioned above qualify.

¹¹In fact, this formula applies to any and-or directed acyclic hyper-graphs, not just hyper-trees. We can therefore use it for the class of DAGs discussed in Appendix B.

¹²This is proven in [GO91, Lemma 3].

¹³In general, each GET-NUMBER_i (resp., FIND-PROBS_i, FIND-OPTDS_i) is a variant of the GET-NUMBER₀ (resp., FIND-PROBS₀, FIND-OPTDS₀) algorithm discussed above. Each PAO_i algorithm is composed on GET-NUMBER_i, FIND-PROBS_i and FIND-OPTDS_i. Notice that some of these modified versions will accept slightly different sets of inputs, or return slightly different results, than the originals.

Different Number of Samples for Different Experiments: Both PAO₀ and PAO₁ (the version that uses GET-NUMBER₁) assume that FIND-PROBS will perform the same number of trials for each probabilistic arc (e.g., for each retrieval). However, we see from the lemmata above that certain arcs require fewer trials than others. In particular, for general hyper-trees, each $d_i \in D$ retrieval requires only

$$m_i = \mathcal{GN}'_{AOG}(d_i, G, \epsilon, \delta) \stackrel{\text{def}}{=} [2 \left(\frac{n C_{\neg}[d_i]}{\epsilon} \right)^2 \ln \frac{2n}{\delta}] \quad (2)$$

trials, using $C_{\neg}[\cdot]$ function defined in Lemma 3.1. Notice that

$$\mathcal{GN}'_{AOG}(d_i, G, \epsilon, \delta) < \mathcal{GN}_{AOG}(G, \epsilon, \delta) = M$$

We now use this observation in two ways. First, we can continue to use the GET-NUMBER₁ algorithm, but identify the class of and-or hyper-trees with the

$$\tilde{M} = \mathcal{GN}_{AOG}(G, \epsilon, \delta) \stackrel{\text{def}}{=} \max_d \{\mathcal{GN}'_{AOG}(d, G, \epsilon, \delta)\}$$

formula, rather than \mathcal{GN}_{AOG} . (Here, we would need only $\tilde{M} \cdot n$ total samples, which is strictly less than $M \cdot n$.)

Second, we can define the PAO₂ algorithm (that uses modified GET-NUMBER₂ and FIND-PROBS₂ subroutines) to perform an even smaller total number of retrievals. Here, GET-NUMBER₂ returns a vector of numbers $\tilde{M} = \langle m_1, \dots, m_n \rangle$, where each m_i is the number of trials required of the d_i database retrieval, defined in Equation 2. The FIND-PROBS₂ algorithm would then perform only m_i retrievals, for each d_i . Hence, this requires only $\sum_{i=1}^n m_i < M \cdot n$ retrievals.

Of course, FIND-PROBS₂ cannot simply use DP⁽¹⁾ (as FIND-PROBS₀ had), as that would force FIND-PROBS₂ to perform the same number of samples for each retrieval. Therefore, FIND-PROBS₂ uses the different (though related) DP⁽²⁾ procedure that only pursues a path if necessary — e.g., it would only follow a path (e.g., $\langle R_{bp}, D_p \rangle$) if it contains a probabilistic arc (e.g., D_p) for which we still need more trials. (E.g., imagine we needed $\mathcal{GN}'_{AOG}(D_c, G_C, \epsilon, \delta) = 25$ trials of the D_c retrieval, but only $\mathcal{GN}'_{AOG}(D_p, G_C, \epsilon, \delta) = 15$ for D_p . Then DP⁽²⁾ would test both D_c and D_p for the first 15 queries given by the oracle; afterwards, for the next 10 queries, it would only test D_c . Hence, it would perform a total of $(15 + 15) + 10 = 40$ retrievals.¹⁴)

(Another variant involves the DP⁽³⁾ procedure that, for each query, stops as soon as it knows the answer. E.g., after seeing that the D_c retrieval succeeded for

¹⁴For comparison, observe that DP⁽¹⁾ would, instead, have performed $\max\{25, 15\} = 25$ trials of both D_c and D_p , for a total of 50 retrievals.

KB_C , DP⁽³⁾ would return success and stop; by contrast, both DP⁽¹⁾ and DP⁽²⁾ might continue exploring the graph, to determine whether D_p was blocked or not. Of course, DP⁽³⁾ would have to vary its strategy over time, to make sure each retrieval gets enough samples — i.e., to avoid having 1000 samplings of D_c but 0 samplings of D_p . Hence, the user could use this DP⁽³⁾ system to answer his queries, efficiently, during PAO's “training mode”. This process is discussed in detail in [GO91].)

4.2 Other Classes of KBs

While many standard derivation systems do correspond to and-or hyper-trees, there are exceptions. We can use the GET-NUMBER₁ algorithm (defined above) to deal with such systems, provided the collection of $\{\mathcal{GN}_G\}$ formulae includes these other classes of graphs.

The problem with most graphs that are more general than and-or trees is in computing the optimal strategy: It is NP-hard to compute the optimal strategy from a context whose structure is a DAG (rather than a tree) and probability vector [Sah74, Gre91b]. Hence, the FIND-OPTDS subroutine cannot simply find the optimal strategy for the graph and the frequency vector (as FIND-OPTDS₀ does, using \mathbf{T}_{AOI} for and-or trees).

Fortunately, there are efficient “approximation algorithms” for certain classes of such contexts. Each such algorithm, $\tilde{\mathbf{T}}_G$, defined for a class of graphs \mathcal{G} , takes a graph $G \in \mathcal{G}$ and probability vector Q (whose i^{th} component is the success probability value of the i^{th} probabilistic arc in G), and returns a complete strategy whose cost is guaranteed to be no more than $\lambda = \lambda(G, Q)$ over the cost of the truly optimal strategy for this graph and probability vector. (Appendix B presents an example of such an algorithm for one class of graphs.) Here, we would first determine whether the given ϵ value is large enough; i.e., if $\epsilon > \lambda$. If so, GET-NUMBER₃ could produce a number, $M' = \mathcal{GN}_G(G, \epsilon - \lambda, \delta)$ that is “ $\epsilon - \lambda$ ” accurate. (I.e., consider the strategy obtained using the (intractable) \mathbf{T}_G algorithm on this graph and the success frequencies obtained from M' trials of each retrieval. Its cost is within $\epsilon - \lambda$ of the cost of the optimal strategy, with probability $\geq 1 - \delta$.) The FIND-OPTDS₃ algorithm would hand these observed frequencies, together with the context, to the approximation algorithm $\tilde{\mathbf{T}}_G$. With probability $\geq 1 - \delta$, the cost of the resulting $\Theta_{pao} = \tilde{\mathbf{T}}_G(G, \hat{P})$ will be within $(\epsilon - \lambda) + \lambda = \epsilon$ of optimal, as desired.

4.3 Handling Other Situations

Range of Queries: So far, we have insisted that each of the queries (i.e., each $q \in Q$) is an instantiation of the root node of the inference tree. We can, however,

allow different queries to correspond to different nodes — e.g., allow some queries to be instantiations of the **BuyCar(κ)** node and others, of **Pretty(κ)**, etc.

We must define the expected cost of each strategy appropriately: For each strategy Θ and each node n in the inference graph, we first define $E^n[\Theta]$ to be the expected cost of using Θ to solve the queries that are instantiations of n ; this is obviously relative to this distribution of queries.¹⁵ We can then define Θ 's overall expected cost to be the weighted average of these “relative” expect costs, weighted by their respective frequencies of occurrence.

Probabilistic Intermediate Arcs: So far, the only probabilistic arcs we have considered correspond to database retrievals (e.g., to D_c , corresponding to **Cheap(κ)**). In general, however, arcs based on rules can also be “probabilistic”. As an example, consider the arc based on the rule $R_{mm} : \text{Man}(x) \Rightarrow \text{Status}(x, \text{Mortal})$, that descends from the **Status(κ, Mortal)** node to **Man(κ)**. Notice this arc is blocked if the query is **Status($\kappa, \text{Immortal}$)**, as this rule would not apply in that situation. ([GO91] discusses solutions to some of the subtle problems that arise from the precedence constraints among the experiments.)

Different Fact Sets for Different Queries: So far, we have assumed that the set of facts associated with the knowledge base is invariant over time. We can, however, allow the database of ground atomic propositions to vary from query to query. (*N.b.*, we are *not* changing the rules that form the inference graph; e.g., R_{bc} and R_{bp} are kept.) For example, the first query (e.g., **BuyCar(C1)**) can be in the context of a knowledge base that contains the two facts **{Cheap(C1), Pretty(C2)}**; the second query (e.g., **BuyCar(C8)**), in the context of the single fact **{Pretty(C8)}**; the third query, (e.g., **BuyCar(C8)**), in the first context; etc etc etc.

In general, we can consider the input to each DP be a pair (q, DB) , where q is a query corresponding to a node of G and DB is a database of ground atomic propositions. Notice each such pair specifies which probabilistic arcs are blocked. All of the analyses shown above will hold, provided there is some fixed probability that each particular arc will be blocked, over the distribution of such pairs, and these events are independent.

Not Just Derivation Systems: While this work is worded in terms of finding the probably approximately optimal *derivation strategy*, this analysis also applies to any situation that involves finding a single satisfying answer within a search space. In general, this anal-

¹⁵Of course, this DPe will only use the portions of Θ that is relevant to this query — i.e., that corresponds to the subgraph that is “under” n .

ysis pertains to any directed hyper-tree where [1] the task involves reducing the root down to some subset of the leaf-nodes; [2] various hyper-arcs can be blocked, with some unknown probability; and [3] each arc has some fixed cost.

Hence, this analysis can be used to deal with a set of Operators that can be used to reduce a Goal down to a set of Primitive Actions, each of which might or might not succeed, etc.

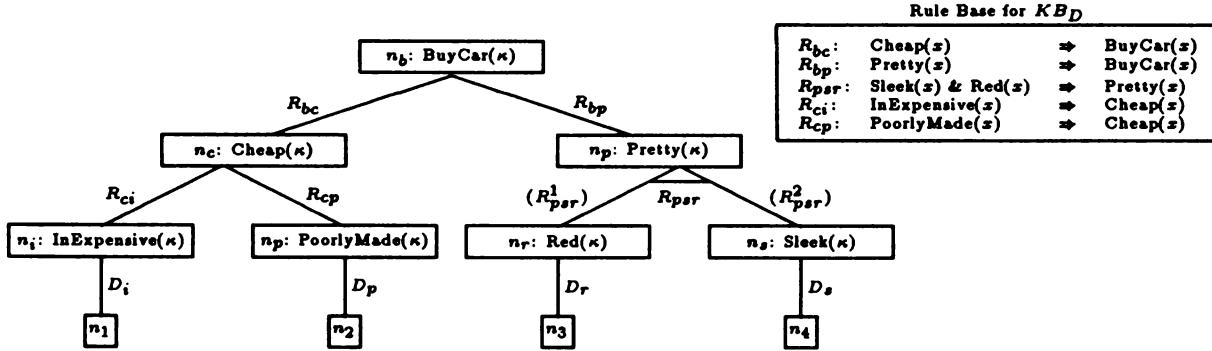
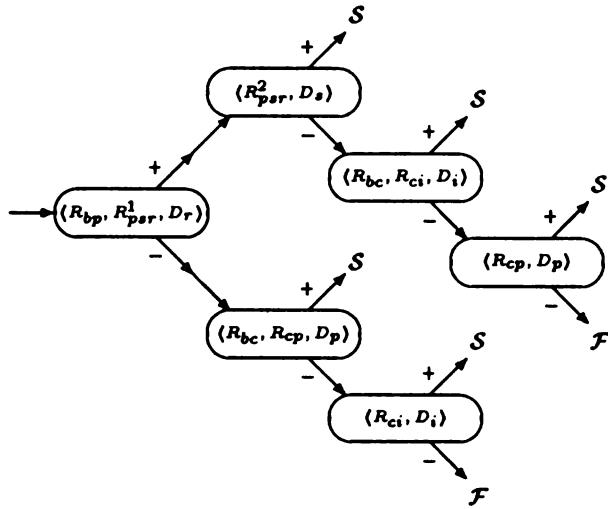
5 Conclusion

The results presented in this paper borrow from, and extend, various other lines of research. It takes its framework, viewing a derivation process as a graph search, from the work on effective use of control information ([SG85], [TG87], [Smi89], etc.). Those works (like the work on satisficing search strategies from which they arose [Gar73, SK75, Bar84, Nat86, Gre91a]) require that the user supply specific probability values for the probabilistic arcs, or use methods for computing these values that are based on unlikely assumptions. Our work fills this gap, by providing an effective, efficient technique for finding good estimates of these values. In addition, most of their techniques insist on finding the “optimal” solution; this becomes problematic when that task is intractable. Our work implements the obvious way of sidestepping this limitation, by allowing near-optimal solutions.

Our approach also resembles the work on EBL (“explanation-based learning”) [DeJ88], as it uses previous solutions to suggest an improved derivation system. Most EBL systems use only a single example to suggest a new strategy; we extend those works by showing how to use a set of samples and by describing, further, the exact number of samples required. While most EBL systems are purely heuristic in nature, we use techniques from mathematical statistics to guarantee that our new strategies will (usually) be close to optimal.

Finally, this work borrows those statistical methods (as well as its title) from the field of “probably approximately correct learning” [Val84]. We provide a concrete application of these theoretical techniques.

To recapitulate: this work describes a technique for improving the efficiency of a derivation system. This process, PAO, uses a computed number of examples of the derivation process’s behavior to determine which derivation paths are likely to succeed, then uses this information to identify a good strategy: one whose expected cost will be arbitrarily close to optimal, with arbitrarily high probability. Within this framework, it completely solves several important, general classes of knowledge bases, including those whose rules form and-or hyper-trees.

Figure 2: And-Or Inference Graph G_D Figure 3: Strategy Tree Θ_D for the And-Or Graph G_D

A And-Or Hyper-Tree Inference Graphs

This paper has focused on disjunctive KBs for pedagogical reasons, as they are much easier to describe. This appendix uses an example of a non-disjunctive definite clause KB to illustrate a non-degenerate hyper-tree, and a strategy for this structure.

Consider the KB_D knowledge base, that includes the rules shown in Figure 2. As above, we can arrange the rules into an inference graph, $G_D = (N_D, A_D, S_D, B_D, c_D)$. Each $n \in N_D$ refers to an atomic proposition to be proven. Here, $A_D \subseteq N_D \times 2^{N_D}$ is a set of *directed hyper-arcs*; meaning an arc can lead from one node to a set of children nodes, where these nodes, conjunctively, imply their common parent. As an example, the R_{psr} arc connects n_p to $\{n_r, n_s\}$. Each member $s \in S$ is a set of nodes; i.e., $S_D \subseteq 2^{N_D}$. (E.g., S_D is $\{\{n_1\}, \{n_2\}, \{n_3, n_4\}\}$.) A derivation process is successful if it reaches each member of some $s \in S$.

As above, a *strategy* specifies the order in which to

consider the various possible arcs. These strategies are much more complicated to describe than the ones for disjunctive KBs (e.g., KB_C). There, a strategy could terminate, successfully, after performing a single successful database retrieval. This is not necessarily true here — i.e., we may have to perform several other successful retrievals before returning Yes. (For example, it is not sufficient for the D_r retrieval alone to succeed; the D_s retrieval must succeed as well.) Hence, each strategy must specify the action to take both when a retrieval succeeds, and when it fails.

The Θ_D “strategy tree” shown in Figure 3 is an example of a strategy for traversing the G_D graph. The derivation process that uses this strategy, DP_{Θ_D} , will first follow the path given in Θ_D ’s initial (i.e., leftmost) node, (R_{bp}, R^1_{psr}, D_r) . Notice the final step, D_r , is probabilistic; if it succeeds, then DP_{Θ_D} will follow the +labeled arc to the (R^2_{psr}, D_s) node in the Θ_D strategy tree; otherwise, it will follow the --labeled arc to (R_{bc}, R_{cp}, D_p) node. Assume that D_r was successful (i.e., that the $Red(\kappa)$ retrieval succeeded); DP_{Θ_D} then follows the instructions on the (R^2_{psr}, D_s) node, meaning it follows R^2_{psr} and attempts the D_s retrieval. If D_s succeeds, DP_{Θ_D} stops, returning success. (This is indicated by the S node, at the end of +labeled arc ascending from Θ_D ’s (R^2_{psr}, D_s) node.) If not, DP_{Θ_D} follows the --labeled arc to (R_{bc}, R_{ci}, D_i) , and so pursues these steps in the G_D tree. Etc etc etc. (See [GO91] for further details of this process.)

We can compute the expected cost of a strategy, given the c cost function and the success probability of each probabilistic arc. We define it recursively, in terms of the cost of the strategy subtree whose root is some node, m . If m is a leaf node (i.e., either S for “success” or F for “failure”), then the cost of m is 0. Otherwise, let m ’s arc sequence be (a_1, \dots, a_q) , where a_q is a probabilistic arc, with success probability $Pr[a_q] = p_q$. (Notice these a_i ’s are arcs in the inference graph, not in the strategy tree.) Let m^+ (resp., m^-) be the node in the strategy tree at the end of the +labeled (resp., --labeled) arc from m . The cost of a (sub)strategy rooted in m is the sum of the costs of the arcs in the

m node (i.e., $\sum_{i=1}^q c(a_i)$) plus p_q times the cost of the sub-strategy headed by m^+ , plus $1 - p_q$ times the cost of the sub-strategy headed by m^- .

(Notice: we need to use this same type of “strategy tree” structure even when considering simple (i.e., non-hyper) inference trees, when intermediate arcs are probabilistic.)

B Efficient Algorithm for finding an Approximately Optimal Strategy

[Gre91b] proves that the task of finding a minimal cost strategy for an arbitrary graph is NP-hard. This reduction proof uses the particular class of DAGs suggested by Figure 4. This appendix provides an efficient algorithm that produces an approximately optimal strategy for this class of structures.

We consider the class of simple 4-deep non-hyper graphs, that have only a single “layer” of nodes with multiple parents — the $\{B_j\}$ nodes. Let $c_b \in \mathbb{R}^+$ be the cost of each $A_i \rightarrow B_j$ arc¹⁶; $c_s \in \mathbb{R}^+$ be the cost of each $B_j \rightarrow S_j$ arc; and $c(i) \in \mathbb{R}^+$ be the cost of each $G \rightarrow A_i$ arc. The only probabilistic arcs are the ones from B_j to the associated S_j success nodes; let p_j be the success probability of the $B_j \rightarrow S_j$ arc.

Notice first that there is an obvious poly time algorithm for computing the optimal strategy if any $p_j = 1$, as the optimal strategy here is simply the minimal cost path from $G(\kappa)$ to the associated S_j success node. (If there are more than one such p_j , we use the smallest of these values.) We can, furthermore, ignore any S_j if its corresponding $p_j = 0$. We can, therefore, deal only with the case where $0 < p_j < 1$ holds for j .

[Gre91b] proves, in general, that the optimal strategy for any DAG is an ordering of the arcs in some embedded tree. It also proves that, for any strategy Θ ,

$$\rho \cdot C_T(T[\Theta]) \leq E[\Theta] \leq C_T(T[\Theta])$$

where $T[\Theta]$ is the tree associated with the strategy Θ (i.e., the tree formed by the arcs used by Θ); $C_T(T)$ (read “the tree cost of the tree T ”) is the sum of the costs of T ’s arcs; and $\rho = \prod_j (1 - p_j)$ is the product of the failure probabilities of Θ ’s probabilistic arcs.

This allows us to bound the cost of any strategy: $E[\Theta] \geq \rho \cdot C_T(T_{opt})$, where T_{opt} is the minimal cost embedded tree. We can use this as a bound for the cost of the optimal strategy Θ_{opt} ,¹⁷ i.e.,

$$C_T(T_{opt}) \leq \frac{E[\Theta_{opt}]}{\rho}$$

Now consider the following algorithm:

¹⁶We let “ $a \rightarrow b$ ” refer to the arc from node a to node b .

¹⁷Notice we are *not* claiming that $T[\Theta_{opt}] = T_{opt}$.

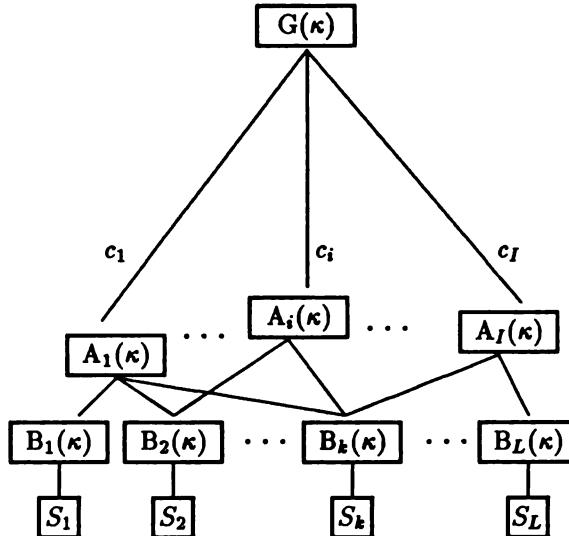


Figure 4: Class of Inference Graphs used in [Gre91b]

$\tilde{T}_{DAG}(G, P)$

- Step 1: Let $K^* = \text{GREEDY}(G)$ be a subset of $\mathcal{A} = \{A_i\}$
- Step 2: Let T^* be the tree built using K^* : T^* ’s nodes are the elements of K^* , G and all of the $\{B_j\}$ and $\{S_j\}$ nodes. T^* ’s arcs are the $G \rightarrow A_i$ arcs for each $A_i \in K^*$; one $A_i \rightarrow B_j$ arc for each B_j ; and all of the $B_j \rightarrow S_j$ arcs.
- Step 3: Let Θ^* be the optimal strategy for T^* , found using [Smi89]’s polynomial algorithm.

This uses the GREEDY algorithm shown below, (a variant of Chvatal’s algorithm for finding an approximately minimal set covering [Chv79]) to find a good enough subset, $K^* \subset \mathcal{A}$. Notice K^* must “cover” the entire $\{B_j\}$ set — i.e., for each B_j , K^* must include at least one A_i such that $A_i \rightarrow B_j$ is in the original DAG.

GREEDY(G)

- Step 0: Set $K^* = \{\}$
For each A_i , let $R_i = R(A_i) = \{B_j \mid A_i \text{ leads to } B_j\}$.
- Step 1: If $R_i = \{\}$ for each i , then stop, returning K^* . Otherwise, find subscript k maximizing the ratio $|R_k|/c_k$.
- Step 2: Add k to K^* , replace each R_i by $R_i - R_k$ and return to Step 1.

Let m be the largest value of $|R(A_i)|$, over all A_i nodes. From [Chv79], we see that this algorithm returns a tree whose tree cost is within a factor of

$$H(m) \stackrel{\text{def}}{=} \sum_{i=1}^m \frac{1}{j} = O(\ln m)$$

of the cost of the optimal subtree.¹⁸ We prove below

¹⁸We can actually get a tighter bound: $C_T(T^*) - C_{bs} \leq$

that this means the cost of $\tilde{T}_{DAG}(G, P)$'s answer, Θ^* , is within the additive constant $\lambda = \left(\frac{H(m)}{\rho} - 1\right) E[\Theta^*]$ over the cost of the optimal strategy. While we can efficiently compute the value of $E[\Theta^*]$, we may prefer to use the (easier to compute) larger bound $\lambda = \left(\frac{H(m)}{\rho} - 1\right) C$, where $C = L(c_b + c_s) + \sum_i c(i)$ is the sum of the costs of all of the arcs in the complete DAG.

Proof: Let Θ_{opt} be the optimal strategy for this DAG; T_{opt} be the minimal cost embedded subtree; and Θ^* be the strategy returned by $\tilde{T}_{DAG}(G, P)$, based on the $T^* = T[\Theta^*]$ subtree.

$$\begin{aligned} E[\Theta^*] &\leq C_T(T[\Theta^*]) \\ &\leq H(m) \cdot C_T(T_{opt}) \\ &\leq H(m) \cdot \frac{E[\Theta_{opt}]}{\rho} \\ &\leq E[\Theta_{opt}] + \left(\frac{H(m)}{\rho} - 1\right) E[\Theta_{opt}] \\ &\leq E[\Theta_{opt}] + \left(\frac{H(m)}{\rho} - 1\right) E[\Theta^*] \end{aligned}$$

The final step uses the fact that $E[\Theta_{opt}] \leq E[\Theta]$ for any strategy Θ , and in particular, for Θ^* . \square .

Acknowledgements

This work was supported by an Operating Grant from Canada's Natural Science and Engineering Research Council to the first author, and by the Academy of Finland to the second. We also greatly appreciate the comments we received from William Cohen, Gerhard Lakemeyer, Dave Mitchell, Dale Schuurmans and Bart Selman.

References

- [Bar84] J. A. Barnett. How much is control knowledge worth?: A primitive example. *Artificial Intelligence: An International Journal*, 22:77–89, 1984.
- [Bol85] B. Bollobás. *Random Graphs*. Academic Press, 1985.
- [Chv79] V. Chvatal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4:233–35, 1979.
- [DeJ88] Gerald DeJong. AAAI workshop on Explanation-Based Learning. Sponsored by AAAI, 1988.
- [DM86] Gerald DeJong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–76, 1986.
- $H(m)[C_T(T_{opt}) - C_{bs}]$, where $C_{bs} = L(c_b + c_s)$ is the cost of the tree from the set of Λ s down to the leaf S s.
- [Gar73] M. R. Garey. Optimal task sequencing with precedence constraints. *Discrete Mathematics*, 4, 1973.
- [GN87] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.
- [GO91] Russell Greiner and Pekka Orponen. Probably approximately optimal satisfying strategies. Technical report, University of Toronto, 1991.
- [Gol79] A. Goldberg. An average case complexity analysis of the satisfiability problem. In *Proceedings of the 4th Workshop on Automated Deduction*, pages 1–6, Austin, TX, 1979.
- [Gre91a] Russell Greiner. Finding an optimal satisfying strategy in an and-or tree. Technical report, University of Toronto, forthcoming 1991.
- [Gre91b] Russell Greiner. Finding the optimal derivation strategy in a redundant knowledge base. *Artificial Intelligence: An International Journal*, forthcoming 1991.
- [Kel87] Richard M. Keller. Defining operability for explanation-based learning. In *AAAI-87*, pages 482–87, Seattle, July 1987.
- [MCK⁺89] Steven Minton, Jaime Carbonell, C.A. Knoblock, D.R. Kuokka, Oren Etzioni, and Y. Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence: An International Journal*, 40(1-3):63–119, September 1989.
- [Min88] Steven Minton. Quantitative results concerning the utility of explanation-based learning. In *AAAI-88*, pages 564–69, San Mateo, CA, August 1988. Morgan Kaufmann Publishers, Inc.
- [MKK86] Thomas M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Example-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.
- [Nat86] K. S. Natarajan. Optimizing depth-first search of AND-OR trees. Technical report, Research report RC-11842, IBM T. J. Watson Research Center, January 1986.
- [OG90] Pekka Orponen and Russell Greiner. On the sample complexity of finding good search strategies. In *Proceedings of COLT-90*, pages 352–58, Rochester, August 1990.
- [Sah74] S. Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4):262–279, 1974.

- [SG85] David E. Smith and Michael R. Genesereth. Ordering conjunctive queries. *Artificial Intelligence: An International Journal*, 26(2):171–215, May 1985.
- [SK75] H. A. Simon and J. B. Kadane. Optimal problem-solving search: All-or-none solutions. *Artificial Intelligence: An International Journal*, 6:235–247, 1975.
- [Smi89] David E. Smith. Controlling backward inference. *Artificial Intelligence: An International Journal*, 39(2):145–208, June 1989.
- [TG87] Richard J. Treitel and Michael R. Genesereth. Choosing orders for rules. *Journal of Automated Reasoning*, 3(4):395–432, December 1987.
- [Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–42, 1984.

Generalizing Prioritization

Benjamin N. Grosof*
 IBM T. J. Watson Research Center
 P.O. Box 704, Yorktown Heights, NY 10598
 Tel.: (914) 784-7100 Internet: grosof@ibm.com

Abstract

We generalize the idea of prioritization that was introduced, in the context of circumscription, by John McCarthy (1986) and formalized by Vladimir Lifschitz (1985). We show that the structure of the prioritization partial order (i.e., the partial order over the set of minimized predicates) was previously restricted in such a way ("layered" or "stratified") as to be unsuitable for many applications of default reasoning, including inheritance. We overcome this by showing how to generalize the prioritization partial order to be *any well-founded* (e.g., finite) strict partial order. We show how to *prioritize arbitrary model-preference criteria, and their syntactic correspondents*. One application of this is to generalize prioritized circumscription. We also develop the concept of *composing* prioritization and show how to prioritize preference criteria expressing internally-prioritized *groups* of minimized predicates (or defaults). We suggest that this enables one to express hierarchical modularity in the specification of default reasoning. Our definition of prioritization is applicable to any kind of preference criteria, even those not about models, as long as they obey the properties of reflexivity and transitivity.

1 Introduction and Summary

The concept of prioritization in circumscription, and related ideas in other non-monotonic formalisms, have received much attention in the AI community recently. There are several intuitions behind the concept of prioritization. Prioritization can be viewed as a tool to

specify confidence information about default beliefs. Alternatively, it can be viewed as a tool to specify semantic, or inferential, precedence. However, a full discussion of the motivations and applications of the idea of prioritization should be a subject of (at least) a whole paper in itself. Rather than repeating the justifications for interest in prioritization, here we will mainly take it as a given. In this paper, we show how to generalize the *formal* idea of prioritization as a method to aggregate preferences.

The concept of prioritization in circumscription was introduced by John McCarthy in (McCarthy 1986). However, he did not give a technically precise definition of prioritization; rather, he gave some examples, and suggested considering partial orders of priorities. Vladimir Lifschitz in (Lifschitz 1985) defined "prioritized circumscription" in detail. However, his definition imposes some restrictions that we find to be undesirably limiting with respect to applications. Firstly, the structure of the prioritization partial order (i.e., the strict partial order over the set of minimized predicates) was restricted there to be what we call *layered* (what some others have called "stratified"). Yet the kind of confidence, or *precedence*, that arises naturally in applications, for example, from the specificity dominance principle in inheritance, is often *not* layered. Here, we show how to generalize the prioritization partial order to be *any well-founded* (e.g., finite) strict partial order, including *non-layered*.

Secondly, both McCarthy and Lifschitz defined prioritization only in the context of minimizing *individual predicates* in circumscription. We show how to *prioritize arbitrary model-preference criteria, and their syntactic correspondents*. One application of this is to generalize prioritized circumscription. We also develop a concept of *composing* prioritization and show how to prioritize preference criteria expressing internally-prioritized *groups* of minimized predicates (or defaults). We suggest that this enables one to express hierarchical modularity in the specification of default reasoning. Our definition of prioritization is applicable to any kind of preference criteria, even those

*This work was begun while in residence at Stanford University, as part of the author's PhD dissertation research.

not about models, as long as they obey the properties of reflexivity and transitivity.

Finally, we discuss how our generalized prioritization relates to similar ideas in several non-monotonic formal systems other than circumscription.

2 What Is Prioritization?

Let us review a classic example of prioritization in default reasoning. Suppose we have two default rules: one says that, typically, *Quakers* are *Pacifists*, and the other says that, typically, *Republicans* are not *Pacifists*. Suppose also that we know that *Nixon* is both a *Quaker* and a *Republican*. Without further information, the two defaults conflict. The *Republican* default tells us to conclude $\neg\text{Pacifist}(\text{Nixon})$, while the *Quaker* default tells us to conclude the opposite. Since we have no way to resolve the conflict, we can conclude neither. However, if, perhaps on the basis of confidence or precedence information, we somehow specify a preference for one default, say the *Republican* default rule, over the other, then we can resolve the conflict: the preferred *Republican* default "wins" and we conclude that *Nixon* is not a *Pacifist*.

We can formalize this in terms of circumscription, cf. (Lifschitz 1985), as follows. Let the "base sentence" B be defined as

$$\begin{aligned} \forall x. \neg ab1(x) &\equiv (\text{Quaker}(x) \supset \text{Pacifist}(x)) \\ \forall x. \neg ab2(x) &\equiv (\text{Republican}(x) \supset \neg \text{Pacifist}(x)) \\ \text{Quaker}(\text{Nixon}) \wedge \text{Republican}(\text{Nixon}) \end{aligned}$$

and minimize the "abnormality" predicates $ab1$ and $ab2$, while letting all the predicates vary. Initially, the predicates are minimized "in parallel", that is, with no strict priority established between them:

$$C(B; ab1, ab2; \text{Pacifist}, \text{Quaker}, \text{Republican})$$

(we follow, in this section only, the notation of (Lifschitz 1985)) which entails neither $\text{Pacifist}(\text{Nixon})$ nor its negation. Then the preference for the *Republican* default over the other can be represented by making the minimization of its corresponding abnormality predicate, $ab2$, have higher priority than the minimization of the other's, $ab1$:

$$C(B; ab2 > ab1; \text{Pacifist}, \text{Quaker}, \text{Republican})$$

which entails $\neg \text{Pacifist}(\text{Nixon})$.

The minimization of each predicate can be viewed as corresponding to a preference criterion on models: the first default corresponds to a preference for models that make more (in the sense of set inclusion) *Quakers* be *Pacifists* (more precisely, for models that have a smaller extension of the predicate

$ab1$, and, equivalently, a larger extension of the wff $\text{Quaker}(x) \supset \text{Pacifist}(x)$); while the second default corresponds to a preference for models that make more *Republicans* be non-*Pacifists*. The prioritization, initially empty, then strict, provides a basis for aggregating the two preference criteria into a single, overall, "global" preference criterion. Initially, that global preference criterion is indifferent between models that make *Nixon* a *Pacifist* and those that make him a non-*Pacifist*. After the strict prioritization is established, though, the latter kind are preferred (strictly) by the global preference criterion. Prioritization resolves the conflict about *Nixon*'s pacifism among the two preference criteria.

Bases for Prioritization Information:

Our intuition is that prioritization is a tool to specify relative confidence or precedence. In considering past and potential applications of prioritization, we have found several kinds of bases of prioritization information.

1. The specificity dominance principle employed in inheritance, e.g., cf. (Touretzky 1986) (Stein 1989), conditional logics, e.g., cf. (Delgrande 1987), and argument systems, e.g., cf. (Loui 1987). Specificity was the basis for prioritization in the motivating examples that (McCarthy 1986) and (Lifschitz 1985) gave when originally developing the idea of prioritization.
2. Reliability of sources. A source might be a communicating agent, or perhaps a sensor. For example, from a child's point of view, "things my mother told me" have precedence over "things my brother told me", which in turn have precedence over "things the school bully told me". (Lifschitz 1988) gives a few examples of reliability as an intuitive basis for prioritization.
3. Recursive depth in programming language semantics: for example, in the semantics of negation as failure in stratified logic programs with negation (Lifschitz 1987b) (Przymusinski 1988).
4. Authority in the legal and organizational senses. For example, federal law takes precedence over state law; policy directives from the head of an organization take precedence over those from subordinates.
5. Temporal precedence. For example, when updating a conventional database, information with a later timestamp typically takes precedence over that with an earlier. Chronological minimization (Shoham 1988) is another example of temporal precedence.
6. Difference in (probabilistically) expected utility of actions based on defaults. For example, an action policy oriented toward an emergency situation has precedence over one oriented toward a routine situation, since in the case when both

apply (i.e., in an emergency), following the first policy but not the second has higher expected utility than vice versa.

3 A Motivating Example

Our generalization of prioritization is motivated by the desire to represent the above kinds of precedence. The previous definition of prioritization, we show, is inadequate.

Figure 1 illustrates an example of “common-sense” knowledge that a worker in a particular company might have. Some of her default beliefs are about meetings. For example, on Tuesdays at 4pm, normally, there is a meeting of her work unit including her boss; however, if the boss is on vacation, then, normally, there is no meeting. Other of her default beliefs are about work attendance. For example, on weekdays, normally, she goes to work. However, normally, if she is sick, then she does not go to work. Likewise, she normally does not go to work when it is a holiday. In addition, she has default beliefs about emergencies, for example that she leaves the building if there is a fire.

In this example, it is natural to think of the precedence information as being defined partly *within* groups of defaults, and partly *between* those groups. When one group takes precedence over a second, every default in the first takes precedence over every default in the second. Thus the partial order of precedence between groups, *composed* with the partial orders of precedence within each group, leads to a partial order of precedence among individual defaults. More generally, we can imagine that groups may be formed of sub-groups. An individual default can be viewed as a primitive group of one.

In our example, the defaults about meetings form one group. Within that, the vacation rule takes precedence over the Tuesday-4pm rule. The defaults about work attendance form another group. Within that, the sick rule and the holiday rule each take precedence over the weekday rule; however, there is no precedence between the sick rule and the holiday rule. Finally, the defaults about emergencies form another group (there might be, say, defaults about what to do in case of earthquakes, violence, heart attacks, etc.; we do not show them to keep this example briefer). The *group* of rules about emergencies takes precedence both over the group of rules about meetings, and over the group of rules about attendance. There is no precedence between the meetings group and the attendance group. Figure 1 shows, as a directed acyclic graph (dag), the partial order of precedence between the groups of defaults.¹ We treat an individual default as a primitive group. Groups are circled. A directed link → indicates

¹Recall that any finite strict partial order is isomorphic to a dag.

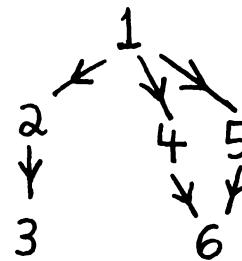


Figure 2: dag view of precedence among the individual defaults in the Meetings example from Figure 1

that the *from* group takes precedence over the *to* group. Figure 2 shows, as a dag, the corresponding partial order of precedence among the individual defaults that results from composition.

Now, suppose we wish to represent the default and precedence information in the above example using the tool of prioritization. There turn out to be two problems with the previous definition of prioritization, given in (Lifschitz 1985). The first problem is that it is impossible to represent the partial order of precedence in our example (Figure 2). Lifschitz’ definition only permits partial orders that are finite and what we call *layered*, i.e., that have a structure similar to the system of rank in the military. (Some others have called this “stratified”.)² Figure 3(a), and the equivalent Figure 3(b), illustrate a layered partial order. Viewed as a dag, a layered partial order consists of a totally-ordered series of one or more levels. At each level, there are one or more elements, with no links between them. Our example (Figure 2), however, is *non-layered*.

The second problem is that Lifschitz defined prioritization only over *individual* minimized predicates (in effect, corresponding to individual defaults). His definition does not permit one to define prioritization directly over *groups* of defaults.

These two difficulties with Lifschitz’ definition arise often. Specificity and reliability, for example, are often not layered. It is natural to specify precedence among groups that are associated with, for example, sources or topics.

²We chose to employ the new terminology “layered” partly because the term “stratified” has previously been used in some of the non-monotonic reasoning literature (e.g., (Brown & Shoham 1989)) in the sense of totally-ordered. Also, we wanted to emphasize the prioritization as given, whereas the usage of “stratified” in the logic programming literature (e.g., (Lifschitz 1987b) (Przymusinski 1988)) involves a non-deterministic choice of a particular stratification given a collection of constraints.

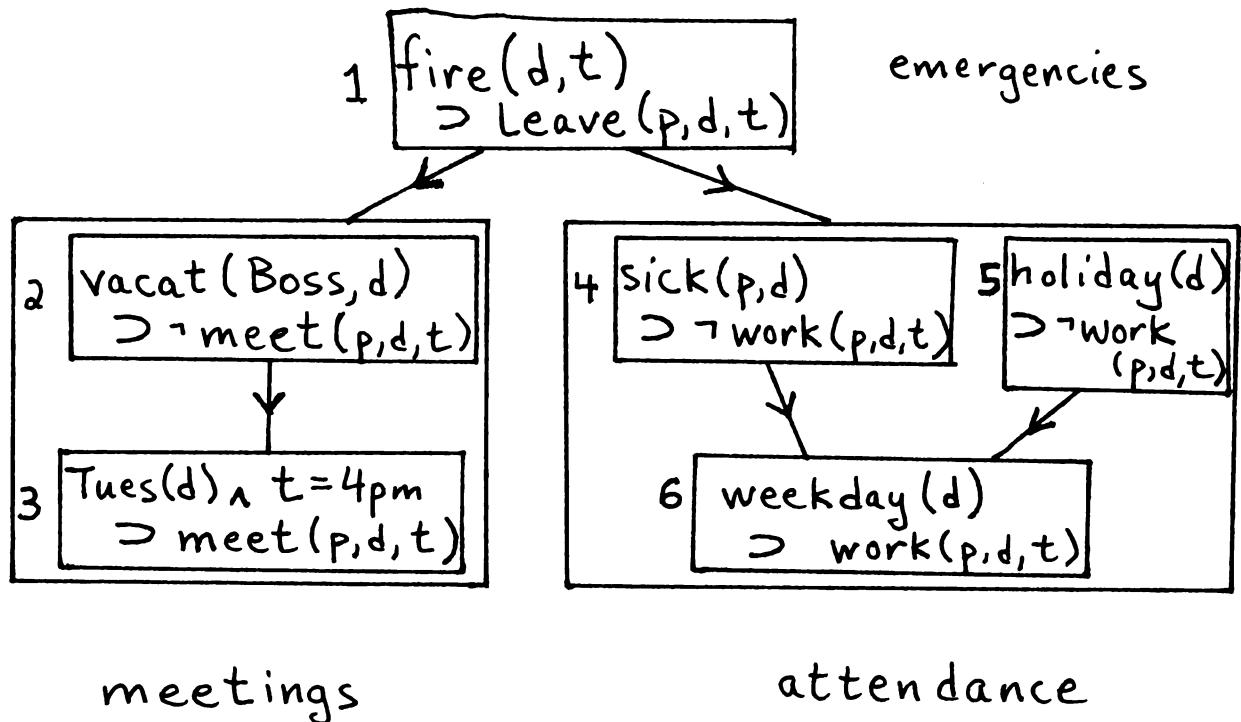


Figure 1: "Meetings" example with non-layered priority

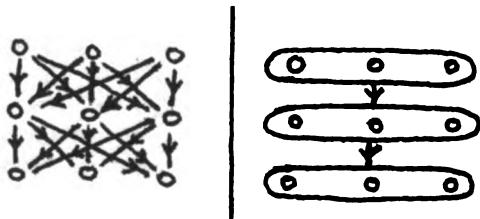


Figure 3: a layered p.o. (a) in detail; (b) via composition

Next, we show how to overcome these difficulties by formally generalizing the idea of prioritization: firstly, to be defined for non-layered partial orders of precedence; and secondly, to be defined (hierarchically) over groups (and groups of groups, etc.) of defaults, i.e., to be composable.

4 Technical Details

Definition 1 (Pre-Order)

A binary relation H is a *pre-order* when it is reflexive and transitive. A non-strict partial order, by contrast, is a pre-order that also obeys anti-symmetry, i.e.::

$$\forall x, y. H(x, y) \wedge H(y, x) \supset x = y .$$

We adopt the following notation for pre-orders. We write $x \leq_H y$ to stand for $H(x, y)$, $x \approx_H y$ to stand for $(x \leq_H y) \wedge (y \leq_H x)$, and $x <_H y$ to stand for $(x \leq_H y) \wedge \neg(x \approx_H y)$. Note that the lack of the anti-symmetry property means that, in general, $x \approx_H y$ does not imply $x = y$.

We say that a pre-order is *defined over a domain \mathcal{D}* when both of its arguments are drawn from that domain.

Viewing a pre-order as a preference criterion or “voting function”, transitivity seems a fairly basic desirable property. Reflexivity means that the preference is non-strict. Lack of anti-symmetry, as well as lack of totality, means that the criterion is quite *partial* in a certain sense: not only is there a class of comparisons for which the “voter” says “it’s a tie”, but also there is another class for which she abstains (“I don’t have anything to say”).

In the context of circumscription, we will be discussing pre-orders that are defined over predicate and function symbols of a “base” logical language \mathcal{L} . In our definitions and applications of circumscription in this paper, we will take \mathcal{L} to be a first-order logical language and define pre-orders that take as explicit arguments a tuple of predicate and function symbols “similar to”, i.e. of the same type as, the tuple of all predicate and function symbols in \mathcal{L} .

Next, we define prioritization as an operation on an indexed set of arbitrary pre-orders, that produces a new pre-order. Prioritization, thus, is a tool to define interesting pre-orders. We will use it to define a generalized idea of prioritized circumscription. In particular, we will consider two kinds of “input” pre-orders to the prioritization operation: predicate pre-orders; and pre-orders that are themselves the product of prioritization.

However, we view prioritization abstractly: as a method to aggregate preference criteria. In our view, it is not tied to the particulars of predicates and circumscription.

Definition 2 (Prioritized Pre-Order)

Let $H \stackrel{\text{def}}{=} \{H_i \mid i \in N\}$ be an indexed set (tuple) of pre-orders, where N is the set (tuple) of indices, and where each H_i is defined over the same common domain \mathcal{D} . We permit any domain: we do not require that it be the models of a logical language, or the symbol tuples (see Z below) corresponding to models. Let R be a strict partial order defined over N , where $R(j, i)$ means that index j has *higher priority* than index i . We call R a *prioritization partial order*. Furthermore, let R be well-founded (in the direction of ascending priority). Note that any (strict) partial order defined over a finite domain is well-founded; therefore, finiteness of N suffices. We define the *prioritized pre-order* $(H; R)$ as:

$$Z \preceq_{(H; R)} Z' \stackrel{\text{def}}{=}$$

$$\forall i \in N. [\forall j \in N. R(j, i) \supset (Z \approx_{H_i} Z')] \supset (Z \preceq_{H_i} Z')$$

The *resulting* prioritized pre-order $(H; R)$ is defined over the same domain \mathcal{D} as the *starting* pre-orders H_i .

You may get some intuition about this definition from the following analogy. View the index $i \in N$ of each starting pre-order H_i as a voter. $Z \preceq_{H_i} Z'$ means that voter i votes non-strictly (i.e., “agrees”) in favor of Z' . Then in order for the overall preference function (i.e., $Z \preceq_{(H; R)} Z'$) to be (non-strictly) in favor of Z' , each voter must vote (non-strictly) in favor in case of a tie (i.e., \approx_{H_j}) by all the voters who are higher-priority ($R(j, i)$) than that voter. Thus, if a voter’s “higher-ups” do not tie, then that voter is “of no account”: his agreement is not required.

Theorem 1 (Well-Definition)

Prioritization according to Definition 2 is well-defined for well-founded prioritization partial orders.

Let R be a well-founded prioritization defined over N .³ Let H be a tuple of pre-orders, indexed by N , each

³Actually, R could be defined beyond N . More generally and precisely, then, let the restriction of R to N be well-founded.

defined over a common domain. Then $(H; R)$ is a pre-order, defined over the same domain.

Proof Overview: Reflexivity of $(H; R)$ is easy to show. However, our proof of the transitivity of $(H; R)$ is non-trivial: it is inductive, using the well-foundedness of R . See Appendix for full proof. \square

The well-foundedness condition is necessary as well as sufficient.

Theorem 2 (Ill-Definition)

Prioritization according to Definition 2 fails to be well-defined when the prioritization partial order is not well-founded.

For any non-well-founded prioritization, there is a starting set of pre-orders for which the result of prioritization according to Definition 2 fails to be transitive. See Appendix for the proof by counterexample.

In (Grosof 1991a, 1991b), we discuss how to modify the definition of prioritization to behave well on a large class of non-well-founded prioritization partial orders, including the counterexample given in the proof.

Definition 2 was defined for arbitrary pre-orders on arbitrary domains. Next, we apply it, in particular, to generalize the definition of prioritized circumscription.

Definition 3 (General Circumscription)

Circumscription in its general form was defined by (Lifschitz 1984) as minimization with respect to a pre-order. Let $B[Z]$ be a formula in \mathcal{L} , where Z is the tuple of all predicate and function symbols in \mathcal{L} . (It will be helpful usually to think of B as a closed formula, i.e., a sentence.) Let H be a pre-order defined over Z . The circumscription of $(Z$ in) $B[Z]$ with respect to H is the (second-order) formula⁴

$$C(B; H; Z) \stackrel{\text{def}}{=}$$

$$B[Z] \wedge \neg \exists Z'. B[Z'] \wedge Z' \prec_H Z$$

Preferences On Models: The essence of the definition of general circumscription is that the *pre-order represents a preference criterion*. For prioritized predicate pre-orders (see below), the preference pre-order formula \preceq_H , defined on tuples Z of all predicate and function symbols corresponds equivalently to a (reflexive, transitive) preference relation \sqsubseteq_H defined on *models* M . The models of $C(B; H; Z)$ are exactly those models of $B[Z]$ that are most preferred with respect to \sqsubseteq_H . The *circumscriptive theory* is defined as the set of sentences satisfied in all models of $C(B; H; Z)$, i.e.,

⁴One can express the fixing of a subset W of the predicates and functions by modifying the minimized pre-order to conjoin the pre-order $W = W'$ to the minimized pre-order. This corresponds to a parallel prioritization.

in all most preferred models of B , following (Lifschitz 1988). In general, however, as (Etherington 1988) remarks, the existence of a model-preference correspondent to a syntactically-defined pre-order is an open question.

Notation: Let $D \leq E$ stand for $\forall x. D(x) \supset E(x)$, where D and E are open formulas with the same arity (i.e., are *similar*), and x stands for a tuple of individual object variables. Let $D = E$ and $D < E$ then be defined analogously to \approx_H and \prec_H above.

The focus of most previous work on circumscription has been on minimizing predicates. Let $P \in Z$ be a single predicate in \mathcal{L} . We define the individual *predicate pre-order* corresponding to P as $\lambda Z, Z'. P \leq P'$. Then, for example, if *flies*, $ab_1 \in Z$, then $\lambda Z, Z'. flies \leq flies'$ and $\lambda Z, Z'. ab_1 \leq ab'_1$ are each an individual predicate pre-order. We saw in section 2 how minimizing a predicate pre-order (e.g., for predicate ab_1) can represent maximizing a default formula (e.g., the rule *Quaker*(x) \supset *Pacifist*(x)).

We define a generalized *prioritized predicate pre-order* $(P^N; R)$, where P^N is a tuple of predicates, indexed by N , as the special case of a prioritized pre-order where each pre-order H_i is an individual predicate pre-order. When there are only a finite number⁵ of predicate symbols in \mathcal{L} , then N must be finite and, therefore, $(P^N; R)$ is well-defined.

We define *generalized prioritized circumscription* as, simply, *circumscription with respect to a prioritized pre-order*, e.g., a prioritized predicate pre-order.

Note that we allow *any* finite strict partial order as prioritization partial order. Our definition of prioritized circumscription thus generalizes the previous definition, given by (Lifschitz 1985). “Prioritized circumscription” as defined there is the special case of prioritized *predicate* circumscription, where the prioritization partial order R is restricted to be finite and *layered* (“stratified”).

Composing Prioritization:

Our definition permits one to *compose* prioritization: the starting pre-orders may themselves be the result of prioritization operations. Thus we can build up a class of prioritized pre-orders recursively or hierarchically. In particular, we can accomplish the goal we discussed in section 3: we can represent a *group* of defaults as one prioritized predicate pre-order, and then represent groups of groups etc. by composing prioritization. It is a theorem that the well-founded class of partial orders is *closed* under (a finite number of) composition operations. In other words, “a prioritization of prioritizations is always equivalent to a prioritization”. By contrast, the layered class is not closed under composi-

tion (e.g. Figures 1 and 2), even when the “composer” partial order is restricted to be parallel, as in Figure 4.

5 Example: Meetings

Next, we show how to use our generalization of prioritization to represent within prioritized predicate circumscription the Meetings example discussed in section 3 and illustrated in Figure 1. As in the *Quaker-Republican* formalization in section 2, let the “base sentence” B_0 comprise the explicit definition of an “abnormality” predicate corresponding to each of the default formulas:

$$\forall p, d, t. \neg ab1(p, d, t) \equiv (fire(d, t) \supset leave(p, d, t))$$

$$\forall p, d, t. \neg ab2(p, d, t) \equiv \neg(vacat(Boss, d) \supset meet(p, d, t))$$

$$\forall p, d, t. \neg ab3(p, d, t) \equiv (Tues(d) \wedge t = 4pm \supset meet(p, d, t))$$

$$\forall p, d, t. \neg ab4(p, d, t) \equiv (sick(p, d) \supset \neg work(p, d, t))$$

$$\forall p, d, t. \neg ab5(p, d, t) \equiv (holiday(d) \supset \neg work(p, d, t))$$

$$\forall p, d, t. \neg ab6(p, d, t) \equiv (weekday(d) \supset work(p, d, t))$$

where the variables p, d, t , should be thought of as varying over persons, days, and times, respectively. Minimizing these abnormality predicates then expresses the maximizing of the corresponding default formulas. We adopt as prioritization R the partial order illustrated in Figure 2. The resulting prioritized predicate pre-order $(ab; R)$ is (we omit the $\lambda Z, Z'$. prefix, where the tuple of all symbols Z is $(ab1, \dots, ab6, fire, leave, \dots)$):

$$\begin{aligned} & [ab1' \leq ab1] \wedge \\ & [(ab1'=ab1) \supset (ab2' \leq ab2)] \wedge \\ & [(ab1'=ab1) \wedge (ab2'=ab2)] \supset (ab3' \leq ab3)] \wedge \\ & [(ab1'=ab1) \supset (ab4' \leq ab4)] \wedge \\ & [(ab1'=ab1) \supset (ab5' \leq ab5)] \wedge \\ & [(ab1'=ab1) \wedge (ab4'=ab4) \wedge (ab5'=ab5)] \supset \\ & \quad (ab6' \leq ab6)] \end{aligned}$$

The overall state of affairs can then be represented via the prioritized predicate circumscription

$$C(B_0; (ab; R); Z)$$

Suppose we continue the example with the for-sure information that a person must be at work to be able to meet, that leaving the building prevents a person from working the rest of the day, that today is Tuesday, and thus a weekday. To represent this, let the base sentence B_1 be the conjunction of B_0 with the axioms:

$$\forall p, d, t. meet(p, d, t) \supset work(p, d, t)$$

$$\forall p, d, t1, t2. (leave(p, d, t1) \wedge (t1 \leq t2)) \supset \neg work(p, d, t2)$$

$$\forall d. Tues(d) \supset weekday(d)$$

$$Tues(Today)$$

where $(t1 \leq t2)$ means time $t1$ precedes time $t2$.

Then we can ask questions about what can be concluded by default about employee Ed. E.g., does he

⁵a common restriction in AI knowledge representation when working with finitely-axiomatized theories



Figure 4: an inheritance example with “columnar” prioritization

have a meeting today at 4pm? The answer turns out to be yes:

$$C(B1; (ab; R); Z) \models \text{meet}(Ed, \text{Today}, 4pm)$$

Suppose next we learn that the boss is on vacation today; let $B2$ be the conjunction of $B1$ with:

$$\text{vacat}(\text{Boss}, \text{Today})$$

Then the default conclusion is that Ed has no meeting after all; the higher-priority default dominates:

$$C(B2; (ab; R); Z) \models \neg\text{meet}(Ed, \text{Today}, 4pm)$$

Next, consider the scenario in which Ed is sick with a cold, instead of the boss being on vacation; let $B3$ be the conjunction of $B1$ with:

$$\text{sick}(Ed, \text{today})$$

In this situation, the Tuesday-4pm default is in conflict with the sick rule, yet there is no strict prioritization specified one way or the other. Maybe the boss is a slavedriver who expects employees for the weekly meeting unless the illness is deadly. Maybe the company policy puts health first. Who knows? The point is we don't. Accordingly, the prioritized theory leaves the question open and entails no conclusion about whether or not Ed has the meeting:

$$\begin{aligned} C(B3; (ab; R); Z) &\not\models \text{meet}(Ed, \text{Today}, 4pm) \\ C(B3; (ab; R); Z) &\not\models \neg\text{meet}(Ed, \text{Today}, 4pm) \end{aligned}$$

Finally, we remark that the overall prioritized (predicate) pre-order $(ab; R)$ can be viewed as the result of composition. The three groups of defaults, about 1)

emergencies (a rather trivial group containing only a single default); 2) meetings (with two defaults); and 3) work attendance (with three defaults) correspond, respectively, to the prioritized (predicate) pre-orders

$$HH1 \stackrel{\text{def}}{=} [ab1' \leq ab1]$$

$$HH2 \stackrel{\text{def}}{=} \{[ab2' \leq ab2] \wedge [(ab2'=ab2) \supset (ab3' \leq ab3)]\}$$

$$HH3 \stackrel{\text{def}}{=} \{[ab4' \leq ab4] \wedge [ab5' \leq ab5] \wedge \\ \quad [(ab4'=ab4) \wedge (ab5'=ab5)] \supset (ab6' \leq ab6)\}$$

(where, again, we omitted the $\lambda Z, Z'$. prefixes on the right hand sides.)

Let RR stand for the prioritization partial order among these three groups: i.e., that the emergency group (group 1) has higher priority than the meetings group (group 2) and higher priority than the work attendance group (group 3). Then

$$(ab; R) \equiv (HH; RR)$$

where HH is the tuple of the HH_i 's. In other words, the overall prioritized pre-order can be viewed as the result of composing the prioritization between the groups (RR) with the prioritizations within the groups.

6 Why Go Beyond Layered?

One may wonder: “Do I really need to make my life more complicated — can't I get away with always just sticking to the layered case of priority?” Of course, in some particular applications, the appropriate precedence information is layered. More generally, one

might try to approximate a non-layered prioritization with a layered one. However, in general, in the context of circumscription, the resulting theory is different: adding a pair (an arc or path in the dag view) to the prioritization relation may result in more conclusions (as more conflict is resolved), while deleting a pair may result in loss of some conclusions. Indeed, for any finite group of prioritized defaults that are logically independent, it is possible to find a base theory B (i.e., a set of for-sure beliefs) that discriminates between any two different given prioritizations. The next two theorems make these observations more precisely.

Theorem 3 (Prioritization is Monotonic)
Increasing the prioritization⁶ of the minimized pre-order is monotonic for any circumscription.

$$\begin{aligned} [\forall xy. R1(x, y) \supset R2(x, y)] &\Rightarrow \\ [C(B; (H; R2); Z) \models C(B; (H; R1); Z)] \end{aligned}$$

where $R2$ is defined over the same indices (e.g., N) as $R1$.

See (Grosof 1991a) for proof.

Theorem 4 (Discrimination)

Let D be a finite tuple of default formulas, indexed by N , in symbols Z , that are satisfiable and logically independent of each other. Let $R1$ and $R2$ be two distinct (different) prioritization partial orders defined over N . Then there exists a satisfiable base sentence $B[Z]$ that results in non-equivalent default theories⁷ for the two prioritizations; i.e., such that

$$C(B; (D; R1); Z) \not\equiv C(B; (D; R2); Z)$$

See (Grosof 1991a) for constructive proof.

Our basic philosophy is that the aim in knowledge representation is to represent what we know — no less,

⁶i.e., adding more pairs (paths) to the prioritization relation (dag)

⁷In order to make the logical independence requirement clearer, we state the theorem here in terms of the case of prioritized circumscription in which default formulas D are maximized directly. This is a generalization of (McCarthy 1986)'s idea of "formula circumscription". Each starting "default" ("formula") pre-order expresses directly a comparison about a default formula, e.g., the pre-order $\lambda Z, Z'. (Quaker(x) \supset Republican(x)) \leq (Quaker(x) \supset Republican(x))$

Essentially, though, you may think of all this as accomplished by minimizing explicitly defined abnormality predicates as we did in section 2 and section 5. Since we anticipated most readers were previously familiar with predicate circumscription, we chose to discuss default reasoning in those terms throughout this paper, even though it is somewhat more direct to employ a formula circumscription style of definition. For the sake of brevity, we will not discuss this in further detail; see (Grosof 1991a) for more.

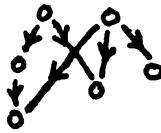


Figure 5: a "random" dag

but also no more. We want the flexibility to represent a given state of precedence information, which may not be layered, and the open-ness to be able to specify later additional precedence information (which might make it become layered). For example, in Figure 1, we would like to be able to leave open the relative precedence of the sick rule and the Tuesday-4pm rule; as we discussed in section 5, they might conflict given the for-sure belief that attending work is necessary to be able to meet. If we later learn that the sick rule has precedence, we can represent that by modifying the prioritization: say, to become $R2$ which includes the extra arc. In our final Meetings example in section 5, then, we will be able to conclude that Ed does not have to meet today:

$$C(B3; (ab; R2); Z) \models \neg \text{meet}(Ed, \text{Today}, 4pm)$$

Inheritance and Specificity:

Even in relatively simple inheritance of the kind found in AI "frame-based" systems (Minsky 1975), the appropriate prioritization is non-layered. Common there is a "columnar" structure: several inheritance chains, each about a different topic, with empty (i.e., parallel) prioritization between the chains. Figure 4 illustrates such a case. Prioritizing in parallel two groups, each of which internally has total prioritization⁸, results in a non-layered overall prioritization.

More generally, in our experience so far in representing specificity dominance in inheritance, we have found that the appropriate prioritization mirrors the directed acyclic graph (dag) of the defeasible links in the inheritance network. Such dags are usually *not* layered. (Recall that any finite strict partial order is isomorphic to a dag.) Figure 5 illustrates a "random" non-layered dag of the kind that arises in inheritance networks (e.g., cf. (Touretzky 1986)).

7 Discussion, Related Work, and Future Work

Many non-monotonic logical formalisms besides circumscription have some analogue of prioritization. One direction of work employs specificity as an implicit

⁸By "total", we mean that the prioritization partial order is a total order.

basis for precedence. Examples of such approaches include inheritance, e.g., cf. (Touretzky 1986) (Stein 1989), conditional logics, e.g., cf. (Delgrande 1987), and argument systems, e.g., cf. (Loui 1987).

One especially interesting relevant piece of work is (Geffner 1989). He interprets defaults as conditional assertions, and develops a theoretical treatment that employs a formalization of precedence based on arbitrary finite (strict) partial orders. The model preference criterion that (Przymusinski 1988) uses to define "perfect" models (as the most preferred with respect to that criterion) is very similar to (Geffner 1989)'s definition. In (Grosof 1991a), we show as a theorem, that the formalizations of precedence in both (Geffner 1989) and (Przymusinski 1988) are each equivalent to a special case of our definition of prioritization (Definition 2).⁹

An interesting direction for future work is to investigate further the relationship between specificity, e.g., in inheritance, and prioritization. The work of (Geffner 1989) is especially promising as a point of departure.

(Ginsberg 1988) defines a formalism based on a bilattice of truth values. His *k* direction in the bilattice corresponds to confidence, and is akin to prioritization. However, since lattices are a special case of partial orders, his approach is fundamentally unable to represent many prioritization partial orders that arise naturally, such as the dags based on specificity in "multiply connected" inheritance networks, e.g., Figure 5.

(Brown & Shoham 1989) define the ideas of "biased logic" and "stratified logic". In our terms, a biased logic is defined via the set of most-preferred models of some standard "base" logic (monotonic: classical or modal) with respect to some particular arbitrary pre-order that is defined over the domain of such models. General circumscription, when the syntactically-defined pre-order has a well-defined corresponding model preference relation, is thus a special case of biased logic. A stratified logic is defined as a biased logic with, in our terms, a model preference pre-order that is defined as the total prioritization of some finite set of starting model preference pre-orders. We defined a more general class of prioritizations, however. Thus our results show immediately how to generalize their idea of a stratified logic. We define a *prioritized logic* to be a biased logic with a model preference pre-order formed via the *arbitrary* (not necessarily total) prioritization of some finite set of starting preference pre-orders. This illustrates how *our idea of prioritization is not tied to circumscription*. Our idea of prioritization is also applicable to other approaches to non-monotonic reasoning based on model preference,

⁹Our definition of prioritization, presented in this paper, was developed independently of their work, at the same time as Geffner's.

including (Selman & Kautz 1989) (Boddy, Goldman, & Kanazawa 1989).

Some other formalisms that define partial orders of degrees of reliability, intuitively about defaults, include (Brewka 1989) (Konolige 1989) (Zadrozny 1987) (Pollock 1987). Each behaves, in general, differently from our definition of prioritization.

Why use prioritization versus one of the other schemes to represent explicit or implicit precedence? The ultimate test is usefulness in application.

More generally, (Doyle & Wellman 1989) suggest that there is a large space of possible methods to aggregate preference criteria for non-monotonic reasoning, and that different methods, e.g., majority voting, may be appropriate in different circumstances. Our perspective is that prioritization is just one tool in the box.

Our concept of prioritization applies beyond the realm of non-monotonic reasoning, to the aggregation of any preference criteria as long as they are pre-orders. One interesting potential direction of application is to formalizing social choice theory and ordinal utility as developed in economics, e.g., in (Arrow 1963).

In (Grosof 1991a), we discuss some desiderata for an aggregation that our generalized prioritization satisfies, including some (about "consensus") that (Doyle & Wellman 1989) propose. We also give some alternative definitions and show that they turn out to be equivalent to our definition.

In (Grosof 1991b), we further generalize our idea of prioritization to represent "pointwise" priorities cf. (Lifschitz 1987a, 1988), and to the case of non-well-founded, infinite prioritization partial orders, e.g., to represent chronological minimization (Shoham 1988) and persistence circumscription (Kautz 1986).

In (Grosof 1991b), we also show how to axiomatize, and infer, circumscriptive prioritization within the same logical language as the defaults, cf. (Lifschitz 1987a, 1988), and how to represent dynamically evolving, i.e., defeasible, prioritization within circumscription.

8 Conclusions

We identified a number of bases for prioritization information. This motivated us to generalize the previous idea of prioritization (due to McCarthy and Lifschitz) in several ways. One is to permit *non-layered* (*non-stratified*) prioritization partial orders, of a kind that arise naturally in many applications: for example, from the specificity dominance principle in inheritance. We showed how to generalize the prioritization partial order to be *any well-founded*, e.g., finite, (strict) partial order.

Secondly, both McCarthy and Lifschitz defined prior-

itization only in the context of minimizing *individual predicates* in circumscription. We showed how to prioritize arbitrary model-preference criteria, and their syntactic correspondents. One application of this was to generalize prioritized circumscription. We also developed the concept of composing prioritization and showed how to prioritize preference criteria expressing internally-prioritized groups of predicates (or defaults). We showed how this enables one to express hierarchical modularity in the specification of default reasoning. Our definition of prioritization is applicable to any kind of model-preference criteria as long as they obey the properties of reflexivity and transitivity. This enabled us to generalize (Brown & Shoham 1989)'s idea of "stratified logics" to a more general category, which we dubbed "prioritized logics".

Our generalized definition of prioritization unifies a number of previous ideas about precedence in the non-monotonic reasoning literature: those of (Geffner 1989) and (Przymusinski 1988), as well as (Brown & Shoham 1989), (McCarthy 1986) and (Lifschitz 1985), are all special cases of our definition. In (Grosof 1991b), we show that the ideas of precedence in chronological minimization (Shoham 1988) and persistence circumscription (Kautz 1986) are also special cases of our definition when well-founded.

A Appendix: Proofs

Line Numbering Notation: An "A" prefix on a line number, e.g., in "(A1)", indicates an Assumption, or premise, or a step working forward from such. A "G" prefix, by contrast, indicates a Goal, or a step working backward from such.

Proof of Theorem 1: We want to show that $(H; R)$ is I) reflexive and II) transitive. We are given that each H_i , for all $i \in N$, is reflexive and transitive.

I) Reflexivity of $(H; R)$ follows easily from the reflexivity of the H_i 's.

II) Transitivity of $(H; R)$

$$\begin{aligned} &\stackrel{\text{def}}{\iff} \forall A, B, C. A \leq_{(H; R)} B \wedge B \leq_{(H; R)} C \supset A \leq_{(H; R)} C \\ &\iff \forall A, B, C. \\ &\quad \forall i. [\forall j. R(j, i) \supset A \approx_{H_j} B] \supset A \leq_{H_i} B \quad (A1) \\ &\quad \wedge \forall i. [\forall j. R(j, i) \supset B \approx_{H_j} C] \supset B \leq_{H_i} C \quad (A2) \\ &\quad \supset \\ &\quad \forall i. [\forall j. R(j, i) \supset A \approx_{H_j} C] \supset A \leq_{H_i} C \quad (G1) \end{aligned}$$

Assume (A1), (A2); it suffices to show (G1).

The quantifiers for indices (e.g., i and j) are understood to range over N , in the rest of this proof.

Note that (A1), (A2) \implies

$$\forall i. [\forall j. R(j, i) \supset A \approx_{H_j} B \approx_{H_j} C] \supset (A \leq_{H_i} B \leq_{H_i} C) \quad (A3)$$

We introduce the following notation:

$$\begin{aligned} D(i) &\stackrel{\text{def}}{=} \{j \mid R(j, i)\} \\ TE^i &\stackrel{\text{def}}{=} A \approx_{H_i} B \approx_{H_i} C \\ OE^i &\stackrel{\text{def}}{=} A \approx_{H_i} C \\ TL^i &\stackrel{\text{def}}{=} A \leq_{H_i} B \leq_{H_i} C \\ OL^i &\stackrel{\text{def}}{=} A \leq_{H_i} C \end{aligned}$$

This is intended to be mnemonic: " D " is an acronym for "Dominator set of"; " TE " is an acronym for "Two Equal", " OL " for "One Less-than-or-equal", etc..

We permit sets, not just singletons, to appear as superscripts. This stands for conjunctions. E.g.:

$$\begin{aligned} TE^{D(i)} &\stackrel{\text{def}}{=} \forall j \in D(i). TE^j \\ OE^{D(k)} &\stackrel{\text{def}}{=} \forall j \in D(k). OE^j \end{aligned}$$

Re-stating: it suffices to show that

$$\forall i. TE^{D(i)} \supset TL^i \quad (A3)$$

implies

$$\forall i. OE^{D(i)} \supset OL^i \quad (G1)$$

By the transitivity of each H_j :

$$\forall j. TL^j \supset OL^j$$

Thus, comparing (A3) and (G1), it suffices to show

$$\forall i. OE^{D(i)} \supset TE^{D(i)} \quad (G2)$$

We prove (G2) by well-founded induction on the partial order R .¹⁰

Let i be an arbitrary index (in N). Our inductive assumption is

$$\forall k \in D(i). OE^{D(k)} \supset TE^{D(k)} \quad (A4)$$

We also assume the left-hand side of (G2):

$$OE^{D(i)} \quad (A5)$$

Then it suffices to show

$$TE^{D(i)} \quad (G3)$$

We use the following lemma:

¹⁰Well-founded induction is a generalized form of mathematical induction that permits any well-founded partial order, rather than only the natural numbers, to play the role of an indexing "sequence". Reviewing it briefly: one assumes that the (parametrized) goal holds for all predecessors (with respect to the partial order) of a given, arbitrary index. Then one tries to show that the goal follows for that index. If successful, it follows by induction that the goal holds for all indices.

$$\forall j. TL^j \supset (OE^j \supset TE^j) \quad (\text{Lemma 1})$$

i.e.:

$$(A \leq_{Hj} B \leq_{Hj} C) \supset [(A \approx_{Hj} C) \supset (A \approx_{Hj} B \approx_{Hj} C)]$$

Proof of Lemma 1: By contradiction.

Assume $(A \leq_{Hj} B \leq_{Hj} C)$ and $\neg(A \approx_{Hj} B \approx_{Hj} C)$. It follows that either $(A \prec_{Hj} B)$ or $(B \prec_{Hj} C)$. But then $(A \leq_{Hj} B \leq_{Hj} C)$ and the transitivity of Hj imply $A \prec_{Hj} C$ which contradicts $A \approx_{Hj} C$.

QED Lemma 1

Comparing (A5) and (G3), using (Lemma 1), \implies it suffices to show :

$$TL^{D(i)} \quad (\text{G4})$$

By the transitivity of R , we have that fact that:

$$\forall k \in D(i). D(k) \subseteq D(i) \quad (\text{A6})$$

$$(\text{A6}), (\text{A5}) \implies$$

$$\forall k \in D(i). OE^{D(k)} \quad (\text{A7})$$

$$(\text{A7}), (\text{A4}) \implies$$

$$\forall k \in D(i). TE^{D(k)} \quad (\text{A8})$$

$$(\text{A8}), (\text{A3}) \implies$$

$$\forall k \in D(i). TL^k \quad (\text{G4})$$

QED Theorem 1

Proof of Theorem 2 :

Our proof is by counterexample. First we show that transitivity fails for a particular non-well-founded prioritization partial order. Then we generalize.

The fundamental intuition behind the counterexample is that Definition 2 leads to a “tie condition” $\approx_{(H; R)}$ that is too weak, i.e., too permissive, when the prioritization partial order is not well-founded.

Let N be the integers (negative as well as positive), representing a discrete timeline, and let each H_i be defined as $\lambda P, P'. P(i) \supset P'(i)$, for some propositional fluent P defined on times $i \in N$. Let R be the total order on the timeline, representing temporal precedence, i.e., $R(i, j) \stackrel{\text{def}}{=} (i < j)$, where here $<$ is ordinary arithmetic less-than. Note that R is not well-founded: it contains an infinite chain in the direction of ascending priority (negative time).

Then the relation $L \stackrel{\text{def}}{=} (H; R)$ is not transitive, as is shown by the following three models of P :

Model	...	-4	-3	-2	-1	0	...
$M1:$...	T	F	T	F	T	...
$M2:$...	F	T	F	T	F	...
$M3:$...	T	F	T	F	F	...

Here T and F stand for *True* and *False*. Before time 0, all three models “tick-tock” back to negative infinity: at each time point $i < 0$, $P(i-1) \equiv \neg P(i)$. Models $M1$ and $M3$ are “in synchrony”; but both are “out of synchrony” with $M2$. At time 0, however, $M1$ and $M3$ differ. Now consider how the relation L compares the three models. $M1 \sqsubseteq_L M2$.¹¹ Why? The key observation is that: if at no index i is the following subformula in the definition of \leq_L satisfied, then the overall pre-order condition \leq_L is satisfied:

$$\forall j \in N. R(j, i) \supset (P(j) \equiv P'(j))$$

Similarly, $M2 \sqsubseteq_L M3$. But $M1 \sqsupset_L M3$, since $M1$ and $M3$ are identical before time 0, and at time $i = 0$, $M1 \sqsupset_{H_i} M3$. Thus transitivity is violated.

This construction of a counterexample extends to any non-well-founded prioritization partial order R . As above, let P be defined on indices $i \in N$, where N is the domain of R . Let the starting pre-orders H_i also be defined as above. By the definition of well-foundedness, R must contain an infinite ascending (with respect to priority) chain of indices. Then, let $M1, M2, M3$ be as above on one such infinite ascending chain, and be extended to be identical to each other everywhere else. Transitivity fails just as it did above. Thus for any non-well-founded prioritization, there is a starting set of pre-orders for which the result of prioritization according to definition 2 fails to be transitive.

QED Theorem 2

Acknowledgements

Vladimir Lifschitz was an invaluable source of advice and discussions. John McCarthy introduced me to circumscription. Thanks also to my adviser Nils Nilsson, Francisco Corella, Hector Geffner, Matt Ginsberg, Yoav Shoham, Kurt Konolige, Joan Feigenbaum, Leora Morgenstern, Andrew Baker, Michael Genesereth, Gerhard Brewka, and Henry Kautz for helpful discussions. The author was supported by a Fannie and John Hertz Foundation Graduate Fellowship while at Stanford University.

References

- [1] Arrow, Kenneth J., (1963) *Social Choice and Individual Values*. Yale University Press, second edition, 1963.
- [2] Boddy, Mark, Goldman, Robert P., Kanazawa, Keiji, (1989) “Investigations of Model-Preference Defaults”. Working paper, Dept. of Computer Science, Brown University, Providence, RI 02912, submitted to *Computational Intelligence*, Sept. 1989.

¹¹For the definition of the \sqsubseteq notation, see the Preference on Models subsection in section 4.

- [3] Brewka, Gerhard, (1989) "Preferred Subtheories: An Extended Logical Framework for Default Reasoning". In *Proceedings of IJCAI-89*, pp. 1043-1049, Detroit, MI., Morgan-Kaufman, 1989.
- [4] Brown, Allen L., Jr., and Shoham, Yoav, (1989) "New Results on Semantical Nonmonotonic Reasoning". In M. Reinfrank et al, ed., *Proceedings of the Second International Workshop on Non-Monotonic Reasoning*. Springer Lecture Notes on Computer Science, Berlin, Germany, 1989.
- [5] Delgrande, James P., (1987) "A First-Order Conditional Logic for Prototypical Properties". In *Artificial Intelligence*, 33:105-130, 1987.
- [6] Doyle, Jon, and Wellman, Michael P., (1989) "Impediments to Universal Preference-Based Default Theories". Technical Report MIT/LCS/TM-416, Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA. 02139, Oct. 1989. Revised and expanded from version in *Proceedings of the First International Conference on Principles of Knowledge Representation*, pp. 94-102, Toronto, Ontario, Morgan Kaufmann, 1989.
- [7] Etherington, David W., (1988) *Reasoning with Incomplete Information*. Los Altos, CA.: Pitman (London) / Morgan Kaufmann, 1988.
- [8] Geffner, Hector (1989) "Default Reasoning: Causal and Conditional Theories". Technical Report 137, Cognitive Systems Laboratory, Dept. of Computer Science, University of California, Los Angeles, CA 190024, Nov. 1989.
- [9] Ginsberg, Matthew L., (1988) "Multivalued Logics: A Uniform Approach to Reasoning in Artificial Intelligence". In *Computational Intelligence*, 4:265-316, 1988.
- [10] Grosof, Benjamin N., (1991a) *Updating and Structure in Non-Monotonic Theories* (working title). Ph. D. thesis (in preparation), Stanford University, 1991.
- [11] Grosof, Benjamin N., (1991b) "Generalizing Prioritization II" (working title), working paper, IBM T.J. Watson Research Center, 1991.
- [12] Kautz, Henry A., (1986) "The Logic of Persistence". In *Proceedings of AAAI-86*, Philadelphia, PA., pp. 401-405, Morgan Kaufmann, 1986.
- [13] Konolige, Kurt, (1988) "Hierarchic Autoepistemic Theories for Nonmonotonic Reasoning". In *Proceedings of AAAI-88*, pp. 439-443, Minneapolis, MN., Morgan Kaufmann, 1988.
- [14] Lifschitz, Vladimir, (1984) "Some Results On Circumscription". In *Proceedings of the first AAAI Non-Monotonic Reasoning Workshop*, pp. 151-64, New Paltz, NY, Oct. 1984.
- [15] Lifschitz, Vladimir, (1985) "Computing Circumscription". In *Proceedings of IJCAI-85*, pp. 121-127, 1985.
- [16] Lifschitz, Vladimir, (1987a) "Pointwise Circumscription". In Ginsberg, Matthew L., ed., *Readings in Non-monotonic Reasoning*. Los Altos, CA: Morgan Kaufman, 1987.
- [17] Lifschitz, Vladimir, (1987b) "On the Declarative Semantics of Logic Programs with Negation". In Ginsberg, Matthew L., ed., *Readings in Nonmonotonic Reasoning*. Los Altos, CA: Morgan Kaufman, 1987.
- [18] Lifschitz, Vladimir, (1988) "Circumscriptive Theories: A Logic-Based Framework for Knowledge Representation". In *Journal of Philosophical Logic*, 17:391-441, 1988.
- [19] Loui, Ronald P., (1987) "Defeat Among Arguments: A System of Defeasible Inference". In *Computational Intelligence*, 3:100-106, 1987.
- [20] McCarthy, John, (1986) "Applications of Circumscription to Formalizing Common-Sense Knowledge". In *Artificial Intelligence*, Vol. 28, No. 1, pp. 89-116, Feb. 1986.
- [21] Minsky, Marvin, "A Framework for Representing Knowledge". In: Winston, P.H., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.
- [22] Pollock, John L., (1988) "OSCAR: A General Theory of Reasoning". Working paper, Dept. of Philosophy, Univ. of Arizona, Tucson, AZ. 85721, March 1988. This is an expanded and revised version of: Pollock, John L., "Defeasible Reasoning". In *Cognitive Science*, 11:481-518, 1987.
- [23] Przymusinski, Teodor, (1988) "On the Declarative Semantics of Deductive Databases and Logic Programs". In: Minker, J., (ed.) *Foundations of Deductive Databases and Logic Programming*. San Mateo, CA: Morgan-Kaufmann, 1988.
- [24] Reiter, Raymond, (1980) "A Logic for Default Reasoning". In *Artificial Intelligence*, Vol. 13, Nos. 1-2, pp. 81-132, 1980.
- [25] Selman, Bart, and Kautz, Henry A., (1989) "Model-Preference Default Theories". Technical Report KRR-TR-89-7, Dept. of Computer Science, Univ. of Toronto, Toronto, Ont., Canada M5S 1A4, April 1989.
- [26] Shoham, Yoav, (1988) *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, Cambridge, MA., 1988.
- [27] Stein, Lynn A., (1989) "Skeptical Inheritance: Computing the Intersection of Credulous Extensions". In *Proceedings of IJCAI-89*, Detroit, MI., pp. 1153-1158, Morgan Kaufman, 1989.
- [28] Touretzky, David, (1986) *The Mathematical Theory of Inheritance*. Morgan Kaufmann, 1986.
- [29] Zadrożny, Włodek W., (1987) "A Theory of Default Reasoning". In *Proceedings of AAAI-87*, pp. 385-390, Seattle, WA., Morgan Kaufmann, 1987.

Naming and Identity in a Multi-Agent Epistemic Logic

Adam J. Grove

Department of Computer Science
Stanford University, CA 94305
grove@cs.stanford.edu

Joseph Y. Halpern

IBM Almaden Research Center
650 Harry Rd, San Jose, CA 95120
halpern@ibm.com

Abstract

Modal epistemic logics for multiple agents often assume a fixed one-to-one correspondence between agents and names. In many situations this assumption is too restrictive. Here, we consider how to allow for agents with several names, agents with no names, and names for groups of agents. We examine some of the difficulties that arise in doing this, particularly the subtleties involving knowledge of one's own identity and the different possible ways of referring to agents. We suggest solutions to the difficulties, and embody them in formal logics, for which we provide complete axiomatizations and (where possible) discuss decision procedures.

1 INTRODUCTION

It is much harder to represent an agent's knowledge about the world when the world contains many agents than it is when the agent is alone. Not only must the agent reason about the state of the world, he must also reason about what other agents know about the world, and what these other agents know about other agents' knowledge about the world, and so on. In practice, any formalism for multiple-agent knowledge is developed using a simpler theory about isolated agents as a basis. Because the presence of multiple agents introduces many new and sometimes subtle issues, the extensions required to the simple (single-agent) theory can be far from trivial. Many of these new issues have in common the themes of *naming* and *identity*, and it is these questions that we examine in this paper.

For the sake of concreteness, we begin by considering a specific (single-agent) epistemic logic, called S5, and the most straightforward extension of this logic for reasoning in the presence of n agents, a logic called S5 _{n} . For many applications, S5 _{n} is appropriate and useful, particularly for analyzing complex systems (see, for example, [Halpern and Moses, 1990;

Rosenschein and Kaelbling, 1986]). However, it is easy to see that such uses for S5 _{n} depend on restrictive conditions about the way agents are named, essentially due to the implicit assumptions that there is a one-to-one correspondence between agents and names and that this correspondence is common knowledge. These assumptions limit the applicability of S5 _{n} . We begin by developing a number of new propositional logics which loosen the restrictions, while retaining many of the desirable features of S5 _{n} (simple propositional syntax, straightforward and intuitive possible-worlds semantics, and a PSPACE-complete validity problem). However, none of our propositional logics escape all the restrictions inherent to S5 _{n} . Attempts to remove these restrictions lead to new and more subtle problems, which we refer to as *relative naming*, *knowledge about oneself*, *scope*, and *multiple ways of referring to agents*. Much of this paper is an introduction to these issues, and a discussion of our solutions to them, which ultimately involve moving to a first-order modal logic with a number of novel features.

We feel that our paper makes two contributions. First, we identify and examine many of the features that make expressive multiple-agent logics much more complex than the familiar single-agent theories. It is true that many of the matters we discuss have been noted before (in, for example, [Dwork and Moses, 1990; Hintikka, 1969; Hintikka, 1986; Kaplan, 1969; Lespérance, 1989; Lewis, 1979; Moses and Tuttle, 1988; Thomason, 1973]). However, our work, which begins by looking at the specific logic S5 _{n} and then examines a series of extensions to it, is a more unified approach to the issues. Second, we present several formal logics—ranging from a straightforward propositional extension to S5 _{n} to a quite expressive first-order extension—to cope with these issues, and give completeness and complexity results for all of them. In this paper we discuss the major issues and results; more details, and proofs of all results, can be found in the full report [Grove and Halpern, 1991].

2 THE LOGICS S5 AND S5_n

2.1 POSSIBLE-WORLDS SEMANTICS

The general ideas that we discuss in this paper are relevant to a broad range of formalisms. However, most of the details we present are based around the standard possible-worlds approach to modal epistemic logic, and on particular propositional modal logics like S5 and S5_n. Here we provide an extremely brief review of these logics and the associated possible-worlds semantics; the reader can find more details and discussion in [Chellas, 1980; Hughes and Cresswell, 1984; Halpern and Moses, 1985].

In all the propositional modal logics we consider, it is assumed that agents reason about the world in terms of some fixed collection Φ of primitive propositions. A formula in the language can be any Boolean combination of the primitive propositions in Φ (formed using \neg , \wedge , \vee , \Rightarrow , and \Leftrightarrow); in addition, for S5_n we allow n modal operators K_1, K_2, \dots, K_n , and so that if φ is a formula, then so is $K_i\varphi$. We read $K_i\varphi$ as *agent i knows* φ .

An S5_n possible-worlds structure (or *Kripke structure*) over Φ is a tuple $M = (W, \pi, K_1, \dots, K_n)$. Here W is a set of states, or possible worlds, and π associates with each possible world a truth assignment to the propositions Φ . More formally, $\pi : W \rightarrow (\Phi \rightarrow \{\text{true}, \text{false}\})$. Each K_i is an equivalence relation on the set of worlds W .

The general intuition here is that a binary relation K_i on worlds can represent agent i 's knowledge, because it can capture his ignorance about what the world is really like. Suppose that the “real world” is represented by $w \in W$. The agent probably doesn't know this (for if he does, there is nothing at all he is uncertain about). On the other hand, some of the other w' will be inconsistent with what the agent knows. So there is some subset of W which contains just the worlds that are consistent with the agent's knowledge. A relation K_i can represent this set: we say that $\{w' \in W : (w, w') \in K_i\}$ is the set of worlds agent i considers possible from w . K_i is sometimes called an (epistemic) accessibility relation. Note that this intuition suggests that K_i can be any binary relation over W . The principal semantic feature which distinguishes S5_n from other modal logics based on the same language are the further conditions that K_i be reflexive, transitive, and symmetric (i.e., an equivalence relation).

Although the properties of S5_n entailed by these conditions have been criticized in the philosophical literature (see [Lenzen, 1978] for an overview), especially the assumption that if an agent doesn't know something he knows that he doesn't know it (see the discussion of axioms in Section 5), S5_n has been shown to be useful and well motivated in the context of an-

alyzing complex systems [Halpern and Moses, 1990; Rosenschein and Kaelbling, 1986]. We briefly review the motivation here. In any possible world, we can suppose that agent i is in a particular *local state*. We imagine the local state as a structure containing all information the agent can reason with and base decisions on. This picture is particularly appropriate for robotic agents where we often can say precisely what memory and sensory capabilities the agent has. Now consider a possible world w , and suppose the agent is in local state s . We can argue that the agent must consider possible *all* worlds, including w , where he is in state s because he has no information to rule any of them out. On the other hand, he should not consider possible any worlds where his state is not s , because the agent can tell that such worlds are inconsistent with what he knows about himself. These arguments show that the accessibility relation partitions the possible worlds according to the state of the agent; thus, the accessibility relation is an equivalence relation.

A formula φ is either true or false at a pair (M, w) consisting of a structure M and a world w in M . We define what it means for φ to be true at world w in structure M , written $(M, w) \models \varphi$, by induction on the structure of φ :¹

- $(M, w) \models p$ (for $p \in \Phi$) if $\pi(w)(p) = \text{true}$
- $(M, w) \models \neg\varphi$ if not $(M, w) \models \varphi$
- $(M, w) \models \varphi \wedge \psi$ if both $(M, w) \models \varphi$ and $(M, w) \models \psi$
- $(M, w) \models K_i\varphi$ if $(M, w') \models \varphi$ for all w' such that $(w, w') \in K_i$.

A formula φ is said to be valid in a structure M , written $M \models \varphi$, if $(M, w) \models \varphi$ for all worlds w in M . Formula φ is said to be (S5_n-) valid if $M \models \varphi$ for all S5_n structures M . In Section 5 we review an axiomatization for the class of all S5_n-valid formulas.

Note that the more familiar logic S5 is simply the special case of S5_n where $n = 1$.

2.2 PROBLEMS WITH S5_n

We have seen that S5_n is a straightforward extension of S5 that can cope with multiple agents. The key step in the generalization to S5_n is that, instead of a single “is known” modal operator K , we allow n modalities K_1, K_2, \dots, K_n . With these, we can form quite complex sentences such as $K_2\neg K_3(K_1\varphi \vee K_2\varphi)$ (agent 2 knows that 3 doesn't know that either 1 or 2 knows fact φ). But, while useful, S5_n has restricted applicability:

- There is a fixed number, n , of agents.
- Each agent has just one name (say agents 1, ..., n , or agents Alice, Bob, Charlie, ...) and an agent

¹The cases for the boolean connectives $\vee, \Rightarrow, \Leftrightarrow$ is omitted because they are definable in terms of \neg and \wedge .

can reason about other agents only in terms of these names (“Alice knows that Bob knows...” but not “Alice knows that someone with red hair knows...”).

- Every name denotes just one agent.
- The composition of the system and the names of the agents are common knowledge, so that every agent knows them, and knows that every agent knows them, and knows that every agent knows that every agent knows them, and so on.

As the following examples show, the reasoning we do about others’ knowledge in everyday life constantly falls outside the limits these assumptions impose.

- There are very many people. No one knows just how many and, in any case, the number changes every second. It would certainly be impossible to know everyone’s name.
- Sometimes we don’t know or care about another’s “true identity”. A customer and salesman in a shop might not bother to ask each other their name. Nevertheless, each will do considerable reasoning about the knowledge and goals of the other. Presumably, each refers to the other using a description or role-name, such as “the salesman”.
- The descriptions we use to refer to others sometimes refer to groups, not single individuals. Further, we frequently reason about the knowledge held by groups of people. This happens even if we are unable to enumerate the names of the people in the group.
- We refer to others in many ways. Sometimes it can be truly surprising to discover that two descriptions or names we have for people actually refer to the same person (consider a costume party as an extreme example of this).
- When we think of names as descriptions, they are clearly not common knowledge. I may not know who “the salesman” really is. Or perhaps I do know this, but he doesn’t know that I know.

Each of these points can be viewed as a question of identity and naming. That is, who are the agents in the world, and how do they refer to each other? If we wish to design agents that operate in multiple-agent domains whose complexity approaches that of the world we encounter every day, then we must accept that these questions have complex answers.

3 PROPOSITIONAL LOGICS

3.1 LOGIC FOR ABSOLUTE NAMING

In general, we see a clear conceptual and practical distinction between the *agents* that exist in a system, and the *names* these agents use to refer to one another. In

the logic S5 this distinction is not made, but in the case of a single agent the confusion is essentially harmless. However S5_n follows the pattern of S5 in failing to keep these two ideas separate, and this does lead to many restrictions as we saw above. It is not difficult to extend S5_n and the possible-worlds semantic model so that the difference between agents and their names is taken seriously. We begin this section by outlining a simple (and still propositional) logic, our *logic for absolute naming*, that does this.²

We start with semantics. A *possible-worlds structure for absolute naming* is similar to an S5_n structure, but has the following differences. First, we extend the semantics to explicitly include a set \mathcal{A} of objects which are the *agents*. As before, each agent $a \in \mathcal{A}$ is associated with an equivalence relation K_a on worlds which models that agent’s knowledge. The agents in \mathcal{A} are semantic objects, and so our model assumes that there is a separate collection of symbols \mathcal{N} , called *names*, which are used to refer to these agents. Whatever the language we use looks like—as we see shortly, there are many useful possibilities—it is these symbols rather than the agents themselves which occur in formulas. Finally, we must say how names are interpreted in the semantics. The following scheme is simple and intuitive, yet fairly general. We suppose that a possible-worlds structure includes a name-interpretation function μ which determines, for each world and each name in \mathcal{N} , some set of agents from \mathcal{A} . We say that $\mu(w, n)$ is the collection of agents with name n in world w . Because we place no restrictions on the function μ , the following are true:

- A name can refer to many agents (for example, “All robots in building 300”). We have also allowed the case where the name denotes no one.
- In any world there can be several names which refer to one particular agent.
- Names are *non-rigid*: their denotation can vary from world to world. This has a number of implications. Most interestingly, if the denotation of name n varies across the worlds that agent a considers possible, then we can reasonably say that a *does not know who n is*. This possibility does not arise in S5_n because the correspondence between name “1” and agent “1” is fixed and therefore known to all agents (in fact, in a sense made precise in [Halpern and Moses, 1990], this correspondence is *common knowledge*).
- It may be the case that agent a is described by a name n which refers to him uniquely and in every possible world. We might say that n is a “proper name” for a . But such proper names need not exist and in a situation with anonymous or interchangeable agents they would not be appropriate.

²The contrast between the absolute and relative names will be discussed in the next subsection.

Our general semantics reduces to the model for $S5_n$ just when every agent has a proper name, and where these are the only names in use.

What language is appropriate for this semantic model? We clearly would like modal operators that allow us to refer to the knowledge of agents with a particular name. The situation is complicated because our names can correspond to groups of several agents, and in general there are many different types of knowledge a group can possess. [Halpern and Moses, 1990] introduces the modal operators S_n , E_n , C_n , and D_n , which can be read as *someone named n knows*, *everyone named n knows*, *it is common knowledge among the agents named n*, and *it is distributed knowledge among the agents named n*, respectively. An assertion φ is common knowledge among the agents n if everyone named n knows φ , everyone named n knows that everyone named n knows φ , everyone named n knows that everyone named n know that everyone named n knows φ , and so on. Roughly speaking, in our semantics a fact φ is distributed knowledge among the agents named n if φ is a consequence of the pooled knowledge of these agents. It is easy to think of many other possibilities, such as “the majority knows...”, which are conceivably useful for some applications.

The number of different ways one can say “a group of agents knows...” already suggests the complications that may arise with multiple-agent formalisms, as compared with theories for one agent. But this issue is somewhat tangential to our more specific concerns about naming, so we do not pursue it in this paper. For simplicity, we focus here on just two of the operators mentioned above, the ones corresponding to “everyone with name n knows” and to “someone with name n knows”. Instead of the n operators K_1, \dots, K_n in $S5_n$, we now have modalities E_n and S_n for each $n \in \mathcal{N}$. We can adopt the convention that whenever n must denote just a single agent, then we write K_n rather than S_n or E_n because the two are equivalent in this case. This shows how our syntax can be regarded as an extension of that for $S5_n$.

The semantic conditions, which define when a formula φ is true at world w in structure M , are straightforward. The new clauses are for formulas of the form $E_n\varphi$ and $S_n\varphi$, and are:

$(M, w) \models E_n\varphi$ if, for all agents $a \in \mu(w, n)$, we have $w' \models \varphi$ for all w' with $(w, w') \in \mathcal{K}_a$

$(M, w) \models S_n\varphi$ if, for some $a \in \mu(w, n)$, we have $w' \models \varphi$ for all w' with $(w, w') \in \mathcal{K}_a$.

We give an axiomatization of the set of valid formulas in Section 5.

In our logic, we can express such assertions as “the teacher knows that everyone in the classroom knows yesterday’s lesson, but that while not everyone knows today’s work there is someone who does”—as

$K_T(E_C p \wedge \neg E_C q \wedge S_C q)$ —and model them semantically. We remark that the two operators we have chosen are likely to be the most useful in practice. It is particularly useful to say “everyone knows” when specifying initial conditions on a system (perhaps, at time 0, every agent knows certain facts about the world). On the other hand, *someone* is good for modeling knowledge acquired as the system evolves. For instance, if a receives a message, it knows that *some* other agent sent the message (so, presumably, also knows its content).

There are many variations on our logic. For instance, we often know something about names like n and m , such as that one always denotes a subgroup of the other, or that n will always denote a single agent, or that m cannot possibly denote the empty set, and so on. Each of these semantic conditions can be characterized by an axiom (in the examples just given, the appropriate axioms turn out to be $(E_m\varphi \Rightarrow E_n\varphi) \wedge (S_n\varphi \Rightarrow S_m\varphi)$, $E_n\varphi \Leftrightarrow S_n\varphi$, and $\neg E_m \text{false}$). In [Grove and Halpern, 1991] these possibilities are explored, along with other much more complex conditions such as requiring that agents “know their own names”. In fact, every one of the assumptions that distinguishes $S5_n$ from our more general logic has simple semantic and axiomatic characterizations, and there is an interesting range of logical theories intermediate between the two.

The ideas behind our logic for absolute naming are straightforward. Indeed, most have been applied before even within the framework of propositional modal logic. A propositional modal logic with non-rigid names was introduced in [Dwork and Moses, 1990; Moses and Tuttle, 1988].³ Some details differ between this logic and ours; in particular, the definition of what it means to say “everyone knows...” is subtly different. Nevertheless, with care our work can be applied directly to the definitions they consider; details are in [Grove and Halpern, 1991]. This is useful because the results we prove about axiomatizations, relationships to other better-known logics, and techniques for the validity decision problem are all applicable. No such results were given (or required) in [Dwork and Moses, 1990; Moses and Tuttle, 1988].

3.2 LOGIC FOR RELATIVE NAMING

The logic for absolute naming can handle a number of important situations that are beyond the scope of $S5_n$; this logic was designed specifically to address the issues we raised in Section 2.2. However, almost as soon as we try to use the formalism to model any actual situation which involves non-rigid names, additional less-expected difficulties arise. In this section, we ex-

³Note that [Dwork and Moses, 1990; Moses and Tuttle, 1988] call non-rigid names *indexical* names. We believe our term is closer to common usage.

amine two of these problems.

The first concerns the need for *relative names*. When we speak, we often identify others relative to ourselves ("the person beside me", "my supervisor", and so on). In this paper, we call these relative names because their denotation is relative to the speaker.⁴ This is a point of practical significance to AI. Lespérance, in [Lespérance, 1989], gives arguments why an agent's knowledge is often limited to identifying objects *relative to himself*, and thus why we should design agents that reason using knowledge of this sort. Thus, we would like to provide a simple (preferably propositional) logic which allows for such relative names.

The second issue relates to the need to be able to model an agent's knowledge about *himself* and who he is. This observation is not new and can be motivated in many ways. For instance, consider that people very frequently use the word "I" as a name to refer to themselves when speaking and reasoning. This suggests that it might be useful to provide a similar construct in a formal system. It is easy to find examples of how this could be useful. For instance, if we want to express an assertion such as "Everyone with name n knows that they (individually) know φ ", using a term like "I" (or "they", "he himself", etc.) saves us from listing all the members of n .

There are other semantic arguments for why we must allow agents to reason about their own identity. In computer science, particularly in distributed systems research and in AI, we sometimes discuss sets of agents which behave sufficiently similarly as to be *anonymous*. That is, they are not programmed to make use of any unique identifier or proper name. Consider a situation where there are just two identically programmed robots, which we call A and B . For this example, we must think of " A " and " B " as something like serial numbers—they are identifiers which we, as external observers, can see stamped on the robots and use in conversation, but which are not available or visible to the machines themselves. It is known by all that, on any particular day, one of the two will be given some command, for instance, to clean the floors in a building. Suppose that today it is A who receives this command. The question is: can we describe this situation in terms of the knowledge held by the robots? Note that it is reasonable to hope to do so; informally, we would surely say that the robots differ in the knowledge they hold. Furthermore, thinking

⁴We should note that a frequently used descriptive term for objects, like names, that are relative to one agent's perspective is *indexical*. We have decided to use "relative" in preference to "indexical" in this paper, first because indexical has also been used with completely different meanings (see, for instance, the previous footnote), and second, because the concept of indexicality sometimes is associated with various philosophical theories and viewpoints that are not necessarily related to our work.

in terms of knowledge is sometimes useful. For example, we might give a very high-level description of the robot's programming in terms of instructions such as: "If you know that you are to clean the floors today, fetch a mop". The interesting question is: just what is it that A knows that B does not? The answer is definitely not that " A cleans the floors today". Neither of them know this, because the agents don't know their serial numbers (as more convincing proof of this, consider that if A was told that " A cleans the floors" this would be in fact be new information). Rather, the answer is that A knows that, from his point of view, "*I* have the command to clean the floors" whereas B knows the negation of this. To formalize this, we must model how agents refer to themselves using names like "*I*" even when there is no more concrete name they can use.

This presents a serious problem for the possible-worlds model as we have been using it. So far, we have been relying on the following apparently reasonable intuitions: (1) possible worlds are models of "the way the world might be" independent of any particular agent's perspective on the world, and (2) an agent's knowledge is modeled as the set of possible worlds consistent with the information he has. It is not hard to see that, under such semantics, A and B have identical knowledge! This is not the conclusion we want to reach, so one of the assumptions must be modified. We suggest—the details are below—that it is best to think of weakening the second assumption. Agents possess knowledge not only about what the world is like, but also about *who they are*.

The two issues just introduced, relative naming and knowledge about oneself, are closely related and share a common solution. Note that standard possible-worlds semantics regards the truth of any formula as being determined by the world alone and ignores the identity of the agent uttering it. Thus, in standard $S5_n$ and in our logic for absolute naming, the truth of a formula is defined with respect to a *world*. But in order to give the semantics of relative names this is not good enough; we must also specify the speaker of the assertion. We can do this formally by defining the truth of a formula with respect to a *pair* (w, a) consisting of world w and an agent a . All relative names are interpreted with respect to a . Formally, there is a name-interpretation function μ that maps worlds and names to binary relations on agents in that world. If $(a, b) \in \mu(w, n)$ then we say that, in w , a calls b by name n . Relative to the pair (w, a) , name n denotes the set $\{b : (a, b) \in \mu(w, n)\}$.⁵

⁵A related extension is to allow relative propositions, so that the function π gives a mapping from proposition symbols, worlds, and agents, to $\{\text{true}, \text{false}\}$. For example, we can have a proposition *leader* which would be true of the pair (w, a) (i.e., $\pi(w, a)(\text{leader}) = \text{true}$) exactly if a is the group leader in world w .

The approach of considering world/agent pairs, which is perhaps the most natural way to give semantics to relative names, also provides a natural way of dealing with knowledge about one's identity. We would say pair (w', a') is considered possible from (w, a) if, at world w , agent a thinks w' is possible (i.e., it is consistent with his knowledge) and also that a might be agent a' in w' . This last clause lets us capture directly a 's knowledge or uncertainty about who he is. Formally, instead of an accessibility relation K_a on possible worlds for each agent a , we have one accessibility relation K which is an equivalence relation on the class of all world/agents pairs. As we have suggested, the intuition for this model is that knowledge cannot be completely represented just as the set of worlds which is considered possible, but also must account for knowledge about who an agent thinks he might be within the world. In our earlier example, A and B would agree on every "objective" assertion about the world (such as, there are two agents, exactly one of them has been told to clean floors, and so on). They differ in who they think they might be: A knows he is the agent that has received the "clean floors" command. This is true even if, as in our example, A is in all other respects identical to B and so has no proper name to use when referring to himself.

Using the ideas just presented, it is easy to extend $S5_n$ to a propositional logic that allows reasoning about relative names and self-identity. The syntax of this logic for relative naming is the same as in our previous logic for naming. We continue to allow an arbitrary collection of names \mathcal{N} . Of course, while the syntax is unchanged, the semantic interpretation of formulas is now based on world/agent pairs as was outlined in the preceding paragraphs. For example, the interpretation of E_n becomes:

$$M, (w, a) \models E_n \varphi \text{ if for all agents } b \text{ such that } (a, b) \in \mu(w, n) \text{ and for all } (w', b') \text{ such that } ((w, b), (w', b')) \in K, \text{ we have } M, (w', b') \models \varphi.$$

According to this clause, any $n \in \mathcal{N}$ is interpreted as a relative name. This is not restrictive, because absolute names, that should not be relative to the agent, are simply the special case where the set denoted is independent of which agent we take as a reference point.

There is one more feature in our logic for relative naming: we assume that the syntax includes a special name I that allows the agent to refer to himself (so that we can say things like "I know that the agent with name n knows ..."). Formally, the name I always has the interpretation of the identity relation on agents with the result that, relative to (w, a) , I denotes just the agent a himself. Thus, $\mu(w, I) = \{(a, a) : a \in \mathcal{A}\}$ for all worlds w . We should emphasize that our name I is not intended to be a formalization of the natural language word ("I"). The best reading of our I depends on context; for instance, we would read $K_n K_m K_I \varphi$

as " n knows that m knows that he himself knows φ ".

In Section 5 we give an axiomatization for this logic. The validity decision problem is $PSPACE$ -complete, precisely the same as that of $S5_n$ [Halpern and Moses, 1985], despite the great increase in expressive power.

Many of the observations we have made appear elsewhere. What is novel is the formulation and investigation of a simple propositional logic which addresses these issues, and our emphasis on why these issues become particularly important in computer science, where the agents can be truly anonymous. Our arguments for "knowing about oneself" are similar to some of Lewis [Lewis, 1979], who gives several other arguments for the significance of this notion and informally introduces semantics which are very similar to those we adopt. Other related earlier work is [Perry, 1979; Castañeda, 1968]. More recently, Lespérance [Lespérance, 1989] has presented a formal logic and semantics incorporating a variant of Lewis's ideas. Briefly, the differences between Lespérance's work and ours are as follows. First, our work (unlike [Lespérance, 1989]) shows how to incorporate "knowledge about oneself" into simple propositional logics, and shows its application for interpreting relative names. In the next section, we give a first-order logic, as does Lespérance. However, we give a general formalism, whereas Lespérance had a more specific and concrete application domain in mind. This leads to many differences. The logic in [Lespérance, 1989] includes a detailed treatment of time and action, very useful and practical issues we do not address. However, Lespérance does not consider at all the problem of "multiple ways of referring", which is our major concern in the next section, as well as being responsible for most of the novel features in our logic. This means that Lespérance's more conventional logic is limited in application to cases where no difficulty arises (as is indeed the case in the example he considers). As well as the differences in semantics, our technical results (on complete axiomatizations and decision procedures) are new.

Finally, we note that the issue of relative naming, particularly for propositional logics, seems to have received little attention. One exception to this is [Moses and Roth, 1989; Roth, 1989] which introduce a language with some simple relative names, and applies the logic to problems in distributed systems research. However, the definition of this logic's semantics is somewhat *ad hoc*; more complex sentences receive unusual (and almost certainly unintended) interpretations. Our semantics, as well as the axiomatizations in Section 5 and the extensions discussed in [Grove and Halpern, 1991], can usefully be applied to the distributed systems applications discussed in [Moses and Roth, 1989; Roth, 1989].

4 FIRST-ORDER LOGICS AND MULTIPLE WAYS OF REFERRING

4.1 MOTIVATION

The propositional logics we have examined seem adequate for reasoning *in the presence of uncertainty about identity*. That is, suppose an agent maintains a knowledge base which consists of sentences in one of these logics. Then he can express and reason using names or descriptions which denote groups of agents, names whose denotations are unknown, names whose denotations are relative to the agent, and so on. However, these logics still have significant weaknesses. First, they are not very good for reasoning *about identity*. To say “I know that the agent in front of me and the robot I repaired yesterday are one and the same” we need to be able to express equality, which moves us beyond the propositional framework of our first logics. It is possible to extend the propositional logics to handle some cases like this, but not all. In many situations we need a richer first-order logic with predicates, functions, and quantification. Other difficulties with our propositional logics are more subtle, although the solution (a first-order logic) is the same. We introduce these issues, *scope* and *multiple ways of referring*, with an example.

Suppose there are two robotic agents, *A* and *B*, and *A* has just broken down. He sends a cry for help over a public broadcast system. *B*, whom we suppose is the agent responsible for dealing with such matters, may or may not have heard. So *A*'s subsequent action depends on whether he can deduce that “I (*A*) know that *B* knows that I need help” (if this is true, he can just wait, but otherwise he should try something else).

So what is a good formalization of “I know that *B* knows that I need help” (that is, what sentence should *A* look for in his knowledge base)? It might seem that something like $K_I K_B(I_need_help)$ would be appropriate. But the situation is really much more complex, because a formula like this can be read many ways.

One possibility is that it could mean that that *A* knows that *B* knows that he (*B*) needs help (the arrangement of quotation symbols in “I know that ‘*B* knows that ‘I need help’” captures what we mean here). We call this the *innermost scope* reading, because the second occurrence of *I* is bound by the innermost modal operator which is K_B . We argue that in many circumstances this is a very simple and useful way to read such sentences. In fact, the propositional logics we consider allow only this interpretation. But this reading is clearly not appropriate in the story above.

One alternative is that the second use of the word or name *I* refers to agent *A*. So we can read the sentence as *A* knows that *B* knows that the agent named *A*

needs help. We call this an *outer scope* reading. The distinction between innermost and outer scopes arises for any non-rigid or relative name. Both readings are useful, yet our propositional logics only capture the former.

However, it is unexpectedly difficult to cope with outer scope in a general way. The problem, which we refer to as *multiple ways of referring*, is that there are usually several reasonable ways to read a name using outer scope. For example, the outer scope reading given in the previous paragraph is not the only possibility. Note that *A* might not have sent a message explicitly saying “*A* needs help”, and instead he might have said simply “*I* need help”. So perhaps *B* didn't know that it was *A* who sent the message. All *B* knows then is that “the agent who just sent a message needs help”. Clearly, this is not so good for *A*. Nevertheless, in some sense *A* is still justified in saying “I know that *B* knows that *I* need help”. After all, “*I* need help” was the content of the message, so if *B* heard it there should be some sense in which *B* knows it, no matter what *B* can infer about the identity of the sender. Here, *B* knows that “the agent who just sent a message” needs help, and it is true that this agent is *A*. The point we wish to emphasize is that *B* might refer to *A* using some description other than the name “*A*”.

For yet another possible (outer scope) interpretation, suppose that *B* can always tell from which room the message was sent, and that there is never more than one agent in any room. Then, again, *A* can say “I know that *B* knows that I need help”. If the assertion is true in the sense that *B* knows that “the agent in room *X* needs help”, where *A* is the agent in room *X*, then *A* need do nothing more. He does not even need to tell *B* where he is. This example is like the previous one: it is still outer scope (the “*I*” refers to *A* somehow), but we must be careful about *how B* refers to *A*. Let us suppose that the robots in this factory are more or less interchangeable (they have similar abilities). Then the agents might identify themselves to each other using the room they are in or the task they are performing. In that case, names like *A* or *B* could be similar to serial numbers: useful to us as external observers for describing what is going on, and yet totally irrelevant to the interaction of the robots among themselves.

There is sometimes an assumption that the “correct” reading of an assertion like “I know that you know that I know...” has you referring to me by my “proper name”.⁶ While referring to an agent by his/her proper

⁶Much more sophisticated theories of such usage of words like “I” have been given in the philosophical literature, for instance [Kaplan, 1977]. Nevertheless, the general motivation seems to be the desire to give a single account of the interpretation of such utterances in natural language. This is definitely not our goal; rather, we seek a theory that helps us formally ascribe “knowledge” to any type of

name is often the appropriate choice, it is not always the appropriate choice. In the situation described in the previous paragraph, the correct way for robots to refer to each other is using position and not using names like A and B . The reading of "I know that B knows that I need help" given here is not just plausible, it is in fact the most natural interpretation there is.

To summarize, when we interpret names or descriptions, there is an issue of *scope* (in the formula $K_I K_B(I_need_help)$, is it A or B who needs assistance?) and of *manner of reference* (which description or name must B use in referring to A for the formula $K_I K_B(I_need_help)$ to be useful?). An expressive formalism should allow us to deal with both issues. In particular, it should allow us to express outer scope readings of a sentence such as "I know that you know that I know" while not allowing outer-scope sentences which contain ambiguity. We should be able to express all possibilities, while making it clear which is intended in every particular instance.

4.2 A FIRST-ORDER LOGIC

Here we present a first-order epistemic logic which can deal with the issues raised in Section 4.1 While our logic has many features of standard quantified modal logics (see [Garson, 1977] for a survey), there are some significant differences. We first describe, briefly, the language and semantics. We then devote the remainder of Section 4 to discussion of the logic and explanation of how it solves the problems just seen.

Our logic assumes there is some fixed first-order vocabulary of constant, function and predicate symbols V . The logic is multi-sorted, with at least the sorts *agent* and *name*. We suppose that there is a distinguished predicate symbol *In* in V which holds between a pair of agents and a name (pairs, because our logic will be for relative names). Intuitively, $In(a, b, n)$ holds if the agent denoted by term a calls b by name n .

Our logic uses possible-worlds semantics, modified as in Section 3.2 to allow for knowledge about oneself and relative names. A possible world is associated with an interpretation over V , so that worlds are no longer propositional models and are instead first-order structures. Formally, for us a *first-order possible-worlds structure with naming* over V is a tuple $M = (W, \mathcal{K}, \pi)$. Here W is the set of possible worlds. The function π maps W to first-order structures; that is, it specifies the domains for all sorts and gives interpretations to the constant, function and relation symbols in V . We note that π plays the analogous role in this first-order case as the propositional π did before. The accessibility relation \mathcal{K} is exactly as in our propositional logic for relative naming; i.e., it is an equivalence relation on world/agent pairs (where the agent, and we aim for maximum expressivity.

agents that exist in a given world will be determined by π). The final condition we place on the semantics is that the domain of the sort *names* must be the same in every worlds; the reason for this requirement is discussed later.

The syntax of the language, with few exceptions, follows all the usual rules for first-order languages. Additionally, there is a modal operator K_t for every term t of sort *agent*. If φ is a well-formed formula, then so long as φ contains no free variables other than those of sort *name*, then $K_t\varphi$ is also well-formed. Finally, there is a distinguished constant *me* of sort *agent*. The semantics—presented below—treat *me* differently from other constant symbols in V in that the interpretation of *me* is not given by the function π .

The final formal definitions concern how formulas are interpreted relative to the semantic model. With few exceptions, the details are tedious but entirely standard, so we omit them. In particular, we note that usual rules for quantifier evaluation apply. In order to allow for knowledge about oneself, formulas are evaluated relative to pairs of a world and an agent. Relative to (w, a) , the special symbol *me* is assumed to have denotation a . That is, *me* plays an analogous role to the name *I* used earlier in that it is the term that allows an agent to refer directly to itself. (The practical distinction between *I* and *me* is minor and technical; the former denotes a name which is essentially the identity relation on agents, while *me* is of sort *agent*.) The semantics for knowledge, i.e., formulas of the form $K_t\varphi$, is very similar to our logic for relative naming. Thus, $(M, w, a) \models K_t\varphi$ if $(M, w', b') \models \varphi$ for all (w', b') such that $((w, b), (w', b')) \in \mathcal{K}$, where b is the denotation of the term t in (w, a) .

In summary, our logic has two relatively novel features as compared to other first-order modal theories. First, we adopt the idea of semantics based on world/agent pairs, so that agents can reason about their identity. The other key difference between our logic and most standard logics is our treatment of *quantifying-in*. Consider a sentence such as "There is a particular agent, who I know needs help" (or, more informally, "I know someone who needs help"). A fairly standard way to express this might be something like $\exists \hat{a} : K_{me} need_help(\hat{a})$.⁷ This formula is said to exhibit *quantifying-in*, because a modal operator (in this case K_{me}) separates a variable from its binding quantifier. Our syntax has been designed to prohibit this. The formula is not well formed since, as mentioned above, a formula of the form $K_t\varphi$ is not well formed if φ has a free variable of sort *agent*. Only quantifying-in for *names* is allowed by our syntax. The motivation for this restriction is discussed in the next section.

⁷Here and below we use \hat{a} to denote a variable of sort *agent*; similarly, we use \hat{n} to denote a variable of sort *name*.

4.3 USING THE LOGIC

Consider the assertion $\exists \hat{a} (K_{me} \text{need_help}(\hat{a}))$ introduced above. We suggested that this might be proposed as a formalization of “I know someone who needs help”. But, following our example in Section 4.1, we know that there is no unique way to read a sentence like this because it doesn’t say *how* the agent is being referred to (by *me*). Am I referring to someone with a particular appearance, in a particular position, with a certain proper name, or something else? Agent quantification across modal operators is useful only if we have in mind, implicitly, a unique preferred way of identifying and referring to individual agents. But this is not always the case.

There is another interesting, although less important, reason for avoiding quantifying-in for agents. In typical applications, the set of agents present will vary from world to world. This is appropriate whenever the composition of the system (who the agents are) is not common knowledge. But, as is well known, there are considerable technical difficulties in defining a logic with quantifying-in if the domain of individuals can vary across worlds. We refer the reader to [Garson, 1977] for a summary of these issues and attempted solutions. By prohibiting such quantification, our logic can continue to use intuitive semantics and a straightforward language. We do allow quantifying-in for the sort *names*, but the domain of names has assumed to be fixed. This is a far more reasonable assumption than assuming a fixed domain of agents would be; see [Grove and Halpern, 1991] for more discussion.

In our logic, we use quantification over names in place of quantifying-in for agents. So we need to show how this is done, and also how the problems we have seen are avoided. Let us reconsider the example of “I know someone who needs help.” Whoever this someone “really” is, I must refer to him somehow; we should make this information explicit. We do this using *names*. Suppose that we are actually referring to the person using their location (relative to us). For instance, if I see someone in the distance in trouble I may refer to “*that person over there*”: I am referring to a particular individual, but not by proper name or by appearance (neither of which I may know). The formalization of this is as follows. Let *Loc* be a predicate on the sort *names*, which is true only of names for relative positions (such as “3 feet in front of me” or “1 foot to the left of me”). Let *In* be the predicate discussed earlier relating agents to their names. Then, somewhat informally, if the name (denoted by) *n* satisfies *Loc(n)* then *In(a, a', n)* is true for agents *a*, *a'* just if *n* is a name of (the position of) agent *a'* relative to agent *a*. Having decided how to remove the ambiguity, we can now express “I know someone who needs help” as:

$$\exists \hat{n} (\exists \hat{a} (\text{Loc}(\hat{n}) \wedge \text{In}(me, \hat{a}, \hat{n})) \wedge K_{me} (\forall \hat{b} (\text{In}(me, \hat{b}, \hat{n}) \Rightarrow \text{needs_help}(\hat{b})))) .$$

Roughly speaking, the formula says that there is a name *n* that describes relative positions and an agent *a* such that agent *a* has name *n* relative to *me*, and I know that all agents *b* with name *n* need help. (Typically, there will be a unique agent with name *n*, but we do not need to make that assumption.)

Quantifying-in over agents has been replaced by quantifying-in over names. Other ways of removing the ambiguity of reference (such as using a different type of description, a proper name, or an absolute location) would lead to a formula which differs principally in the use of another predicate than *Loc*. Our formula seems lengthy, but this is precisely because there are numerous other ways the original assertion could be interpreted. Any formalism which distinguishes between all the different possibilities will be less concise than one which permits ambiguity.

Our logic is also expressive enough to cope with the issue of scope that we saw earlier. In short, in a first-order logic all constant symbols are automatically given *innermost* scope. However, our language includes equality and so we can use quantification to bind the value of constants to variables, and in the case of *names* quantifying-in allows us to use these variables in deeper contexts (i.e., inside further modal operators). For instance, the innermost scope reading of “I know that the president of the U.S. is a Republican” could be formalized as $K_{me} \text{Rep}(\text{President})$. This assertion can be true even if know nothing more about who the president is; the alternative is to use an outer scope reading. In this case, we might get

$$\exists \hat{n} [PName(\hat{n}) \wedge K_{me} (\forall \hat{a} (\text{In}(me, \hat{a}, \hat{n}) \Rightarrow \text{Rep}(\hat{a}))),$$

where *PName* is true of *names* (in our technical sense) that correspond to peoples’ actual proper names (such as, “George Bush”). We note that issue of expressing various scopes is solved as soon as any form of quantifying-in is allowed. But if we had permitted this for the sort *agents*, we would be unable to introduce outer scope without simultaneously introducing the “ways of referring” ambiguity we wish to avoid.

The questions of multiple ways of referring and scope, and many of the points we make about quantified modal logic, have been raised elsewhere (see the discussion below for details). The most novel features of our presentation are the following. First, we emphasize that the “ways of referring” problem and difficulties with quantifying-in are particularly important in computer science domains. It might be argued that there really is a unique and privileged way we use to identify other people; but it is easy to imagine scenarios involving robots or computers where this clearly not true. Second, our device of introducing a sort *names* derives much of its utility because of our framework of relative naming and knowing about oneself. Our claims that we always refer to others using names, and that this should be made explicit, would be less convincing

if we only allowed absolute reference. Third, we have presented a formal logic incorporating all these ideas. The semantics are simple and intuitive, and we have given an axiomatization for the class of valid formulas.

Our concerns about scope relate back to work of Russell ([Russell, 1905; Russell, 1919]). Additionally, there is a large philosophical literature on the distinction between *de re* and *de dicto* reference, which correspond closely to our outermost and innermost scopes respectively. Typical concerns here seem to be delimiting the circumstances under which each is appropriate, and deciding which of the two is a better model of natural language usage in various situations. These are important (and occasionally controversial) questions; but not directly relevant to our concern with finding an adequate formalism for expressing all possibilities.

The *de re/de dicto* issue is also connected closely with the “ways of referring” question, because there is a sense in which *de re* usage depends on (in our terminology) there being a single or preferred way of referring to individuals. Another perspective on this is that there should be a single criterion of “cross-world identity” or one “mode of individuation”. Hintikka and Thomason ([Hintikka, 1969; Hintikka, 1986; Thomason, 1973]) have both argued that this is not necessarily the case, and have presented logics and semantics for a more general situation. For example, Hintikka argues that we often refer to objects and people both in a descriptive manner and by means of position. Neither Hintikka nor Thomason share our emphasis on coping with multiple and varying ways of referring, or give semantics that deal with relative names and knowing about oneself in the manner we do. There are numerous other smaller but still significant differences; see [Grove and Halpern, 1991].

Finally we note that the interpretation of quantifying-in that we use, as being quantification over descriptions or names, seems to have appeared first in [Kaplan, 1969] and Kaplan’s motivation appears similar to ours. Our previous remarks about our concern with coping with all situations rather than finding one “best” way of referring, our semantics for relative reference, and the detail to which we have developed our logic, still apply with regard to comparison with Kaplan’s work.

5 AXIOMATIZATIONS

This is a fairly technical section which contains sound and complete axiom systems, and discusses the complexity of the validity decision problem, for the logics we have discussed.

We begin with the relatively well-known results about $S5_n$ (see for example, [Halpern and Moses, 1990]). The following axiom system is sound and complete, with respect to the class of $S5_n$ possible-worlds structures.

- A1. All instances of propositional tautologies
- A2. $K_i\varphi \wedge K_i(\varphi \Rightarrow \psi) \Rightarrow K_i\psi$
- A3. $K_i\varphi \Rightarrow \varphi$
- A4. $K_i\varphi \Rightarrow K_iK_i\varphi$
- A5. $\neg K_i\varphi \Rightarrow K_i\neg K_i\varphi$
- R1. From φ and $\varphi \Rightarrow \psi$ infer ψ (*modus ponens*)
- R2. From φ , infer $K_i\varphi$ (*knowledge generalization*)

That is, the set of formulas that can be generated by this system is exactly the collection of formulas valid in all $S5_n$ possible-worlds structures. It is also possible to find a polynomial-space decision procedure for validity; in fact, this decision problem is *PSPACE*-complete if $n > 1$ (see [Halpern and Moses, 1985]).

It is interesting that the axioms A3, A4, A5 each correspond, in a precise technical sense, to semantic constraints on the knowledge accessibility relations K_i . For instance, A3 is sometimes called the *knowledge axiom*, because it enforces the property that if something is known then it must be true. The semantic counterpart to this is the requirement that K_i be reflexive (so the “real” world is always one of those the agent considers possible). Neither the axiom nor the condition of reflexivity are necessarily appropriate for other attitudes such as *belief*.

The first new logic presented in this paper was our *logic for absolute naming*. Consider the following collection of axioms (as well as A1 and R1 from above). These axioms and rules hold for every $n \in \mathbb{N}$.

- S1. $S_n\varphi \Rightarrow \varphi$
- E1. $E_n\varphi \wedge E_n(\varphi \Rightarrow \psi) \Rightarrow E_n\psi$
- E2. From φ infer $E_n\varphi$
- C1. $S_n\varphi \wedge E_n(\varphi \Rightarrow \psi) \Rightarrow S_n\psi$
- C2. $\neg E_n \text{false} \Rightarrow S_n \text{true}$

Then the following are true (proofs are in [Grove and Halpern, 1991]).

Theorem 5.1: *The above axiom system is a sound and complete with respect to the class of all possible-worlds structures for absolute naming.*

Theorem 5.2: *The problem of deciding whether a formula is valid with respect to the class of structures with absolute naming is PSPACE-complete.*

The next logic we presented was our *logic for relative naming*. Consider the following axioms:

- K1. $E_I\varphi \Leftrightarrow S_I\varphi$ (and so we write either as K_I)
- K2. $S_n\varphi \Rightarrow S_n(\varphi \wedge K_I\varphi)$
- K3. $\neg S_n\varphi \Rightarrow E_n \neg K_I\varphi$
- K4. $K_I\varphi \Rightarrow \varphi$.
- S1'. $\neg S_n \text{false}$.

Most of these axioms have a counterpart in the system S_{5n} . As one example, consider K3. To make our comparison more apparent, suppose that n is a name which always denotes just one agent (so that we can write K_n rather than E_n or S_n). In this case, K3 reduces to $\neg K_n \varphi \Rightarrow K_n \neg K_n \varphi$. This is similar to the axiom A5 in the logic S_{5n} which would be $\neg K_n \varphi \Rightarrow K_n \neg K_n \varphi$. Axiom A5 is often called the axiom of negative introspection (if you don't know something, you know that you don't know it). We can now see that axiom A5 only really matches this intuition about "introspection" if agent n does in fact know that he is n (i.e., he knows his name).⁸ Our new axiom K3 is better because it is valid even without this assumption.

The following results can be shown:

Theorem 5.3: *The system A1, R1, E1, E2, C1, C2, K1, K2, K3, K4, S1', is sound and complete with respect to the class of all possible-worlds structures for relative naming.*

Theorem 5.4: *The problem of deciding whether a formula is valid with respect to the class of possible-worlds structures for relative naming is PSPACE-complete.*

Finally, we present a sound and complete axiomatization of the valid formulas in our first-order logic. In addition to A1 and R1, we include axioms that express sound patterns of classical, first-order reasoning:

Q1. $\forall \hat{x} \varphi \Rightarrow \varphi[t/\hat{x}]$ if t is substitutable for \hat{x} in φ .

R3. From $\varphi \Rightarrow \psi$ infer $\varphi \Rightarrow \forall \hat{x} \psi$ if \hat{x} does not occur free in φ .

Here, \hat{x} is any variable, and φ and ψ can be any formulas in \mathcal{L} . In Q1, $\varphi[t/\hat{x}]$ is like φ , except that all free occurrences of \hat{x} are replaced by t . Just as with classical (non-modal) logic, we must be careful about when such substitutions are performed. For the purpose of presenting these axioms, we say that an occurrence of a term t in formula φ is *free* if (1) no variable appearing in t is bound by any quantifier (for that occurrence of t), and (2) unless t is a variable of sort *name*, the occurrence of t is outside the scope of all K operators in φ . Then we say term t' is substitutable for an occurrence of t in φ if t and t' are of the same sort, and if the both the original occurrence of t , and the occurrence of t' that would result after replacing t by t' , are free in φ . We say that t' is substitutable for t in φ if t' is substitutable for all free occurrences of t .⁹

⁸A similar observation was made in [Hintikka, 1962].

⁹The restrictions on substitution to do with variables not being bound by quantifiers are required for the same reasons as in classical logic. The additional restriction, that prevents substitution within the scope of a modal operator unless both terms involved are variables of sort *name*, is needed because terms can be non-rigid. For example, $\forall \hat{n} (P(\hat{n}) \Rightarrow K_n P(\hat{n})) \Rightarrow (P(N) \Rightarrow K_n P(N))$, where N is a name constant, is not sound for our semantics. In any model where the set of names satisfying P does not vary

In addition to rules for first-order reasoning, we require axioms to deal with equality.

- I1. $\hat{x} = \hat{x}$
- I2. $t_1 = t_2 \Rightarrow (\varphi \Leftrightarrow \varphi(t_2/t_1))$
- I3. $(\hat{x} = \hat{y}) \Rightarrow K_t(\hat{x} = \hat{y})$ if \hat{x} and \hat{y} are of sort *name*
- I4. $(\hat{x} \neq \hat{y}) \Rightarrow K_t(\hat{x} \neq \hat{y})$ if \hat{x} and \hat{y} are of sort *name*.

Again, I1 and I2 are just like the classical axioms aside from our adoption of stronger conditions on "substitutability". In I2, $\varphi(t_2/t_1)$ can be any formula which is like φ but with zero, one, or more substitutable occurrences of t_1 replaced by t_2 .

Finally, there are axioms dealing with knowledge. It turns out that the necessary axioms and rules of inference are similar to those of the system $S5$:

- M1. $K_t \varphi \wedge K_t(\varphi \Rightarrow \psi) \Rightarrow K_t \psi$
- M2. $K_t \varphi \Rightarrow \varphi[t/me]$ if t is substitutable for me .
- M3. $K_t \varphi \Rightarrow K_t K_{me} \varphi$
- M4. $\neg K_t \varphi \Rightarrow K_t \neg K_{me} \varphi$
- R2. From φ infer $K_t \varphi$, if $K_t \varphi$ is a well-formed formula.

In these axioms, t can be any term of sort *agent*. Also, φ and ψ are restricted by the requirement that the resulting instance be a well-formed formula in \mathcal{L} (for example, in M3 formula φ must be closed or only have free variables of sort *name*, because otherwise $K_t \varphi$ would not be well-formed). In M2, only occurrences of me which are outside the scope of every K modality in φ are replaced by agent term a .

Theorem 5.5: *The axiom system described above is sound and complete with respect to our first-order logic of naming.*

Since first-order logic is not decidable, and our first-order logic of naming extends first-order logic, there can be no decision procedure for deciding validity. Since there is a complete axiomatization for the logic, it follows that the set of validity formulas is r.e. (recursively enumerable).

6 CONCLUSIONS

We conclude by reiterating what we feel are the three main points of the paper:

- Strong assumptions about names and reference are often made in logics for many agents; it is important to be aware of when this is being done, and whether the assumptions are necessary or appropriate.

from world to world the antecedent is certain to be true. Because N might denote such a name in some worlds but not in others, $P(N) \Rightarrow K_n P(N)$ could still be false.

- An expressive logic of knowledge should allow an agent to reason about relative names and about its own identity. Only in very special situations—if there is a fixed, finite number of agents each of which has a unique and commonly known name—can we avoid such reasoning. We have given propositional logics that are widely applicable.
- Questions of scope, reference, and quantifying-in, are very subtle. The principal danger is to assume that there is one unique and automatic *way of referring to agents*. This is not generally true: we need to be careful about how reference to agents is performed (which names are used). Our quantified logic provides one way of expressing general assertions about multiple-agent knowledge while ensuring that all ambiguity has been resolved.

Acknowledgments

The authors are grateful to Ron Fagin, Daphne Koller, Hector Levesque, Karen Myers, Yoav Shoham, Moshe Vardi, and particularly Yves Léspérance, for comments and advice relating to this work. The first author was supported by an IBM graduate fellowship.

References

- [Castañeda, 1968] H.-N. Castañeda. On the logic of attributions of self knowledge to others. *Journal of Philosophy*, LXV(15):439–456, 1968.
- [Chellas, 1980] B. F. Chellas. *Modal Logic*. Cambridge University Press, 1980.
- [Dwork and Moses, 1990] C. Dwork and Y. Moses. Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation*, 88(2):156–186, 1990.
- [Garson, 1977] J. W. Garson. Quantification in modal logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic, Vol. II*, pages 249–307. Reidel, 1977.
- [Grove and Halpern, 1991] A. J. Grove and J. Y. Halpern. Naming and identity in a multi-agent epistemic logic, to appear. Technical report, 1991.
- [Halpern and Moses, 1985] J. Y. Halpern and Y. Moses. A guide to the modal logics of knowledge and belief. In *Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 480–490, 1985.
- [Halpern and Moses, 1990] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. An early version appeared in *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing*, 1984.
- [Hintikka, 1962] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.
- [Hintikka, 1969] J. Hintikka. Semantics for propositional attitudes. In *Models for modalities*. Reidel, 1969.
- [Hintikka, 1986] J. Hintikka. Reasoning about knowledge in philosophy: the paradigm of epistemic logic. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference*. Morgan Kaufmann, 1986.
- [Hughes and Cresswell, 1984] G. E. Hughes and M. J. Cresswell. *A Companion to Modal Logic*. Methuen, 1984.
- [Kaplan, 1969] D. Kaplan. Quantifying in. *Synthese*, 19:178–214, 1969.
- [Kaplan, 1977] D. Kaplan. Demonstratives. Unpublished manuscript, UCLA, 1977.
- [Lenzen, 1978] W. Lenzen. Recent work in epistemic logic. *Acta Philosophica Fennica*, 30:1–219, 1978.
- [Léspérance, 1989] Y. Léspérance. A formal account of self-knowledge and action. In *Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 868–874, 1989.
- [Lewis, 1979] D. Lewis. Attitudes de dicto and de se. *Philosophical review*, 88(4):513–543, 1979.
- [Moses and Roth, 1989] Y. Moses and G. Roth. On reliable message diffusion. In *Proc. 8th ACM Symp. on Principles of Distributed Computing*, 1989.
- [Moses and Tuttle, 1988] Y. Moses and M. R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3:121–169, 1988.
- [Perry, 1979] J. Perry. The problem of the essential indexical. *Noûs*, 13:3–21, 1979.
- [Rosenschein and Kaelbling, 1986] S. J. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference*, pages 83–97. Morgan Kaufmann, 1986.
- [Roth, 1989] G. Roth. Message diffusion in anonymous distributed systems. Master's thesis, Weizmann Institute of Science, 1989.
- [Russell, 1905] B. Russell. On denoting. *Mind, N.S.*, 1905.
- [Russell, 1919] B. Russell. *Introduction to mathematical philosophy*. Allen & Unwin, 1919.
- [Thomason, 1973] R. H. Thomason. Perception and individuation. In M. Munitz, editor, *Logic and ontology*, pages 261–285. New York University Press, 1973.

A Temporal Probability Logic for Representing Actions

Peter Haddawy

University of Illinois

Department of Computer Science and Beckman Institute

405 North Mathews

Urbana, Illinois 61801

haddawy@cs.uiuc.edu

Abstract

To operate intelligently in the world, an agent must reason about the consequences of its actions. The consequences of an action are a function of both the state of the world and of the action itself. Many aspects of the world are inherently stochastic, so a representation for reasoning about actions must be able to express chances of world states as well as indeterminacy in the effects of actions and other events. This paper presents a first-order, future-branching temporal probability logic for reasoning about actions. The language allows quantification over time points as well as over domain individuals. The logic can represent the probability that facts hold and events occur at various times. It can represent the probability that actions and other events affect the future. It can represent concurrent actions and conditions that hold or change during execution of an action. The model of probability relates probabilities over time. The logical language integrates both modal and probabilistic constructs and can thus represent and distinguish between possibility, probability, and truth. Several examples illustrating the use of the logic are given.

1 Introduction

Intelligence requires the ability to reason about action. Actions may be performed to affect the state of the world, to affect the state of knowledge of the performing agent, or simply for their own sake. The present work focuses on the first type of action. To choose appropriate courses of such action, an agent must reason about the state of the world, the way actions affect the world, and the way the world state influences the effects of actions.

Many aspects of the world are inherently stochastic, so a representation for reasoning about actions must be

able to express chances of world states as well as indeterminacy in the effects of actions and other events. For example, the statement "If I were to smoke, I would contract lung cancer some years down the road" can at best be said to hold with high likelihood. Uncertain environmental factors can influence my chance of contracting cancer as can uncertainty in the effects of smoking.

Reasoning about actions requires the ability to reason about time. Facts tend to be true for periods of time and events occur at particular times. Actions comprising a plan may occur sequentially or concurrently. Actions and other events affect the future, but not the past. Chance evolves with time: the chance of rain tomorrow may not be the same now as it will be tonight. Ambiguities in the world are resolved with time: before a fair coin is flipped the chance of heads is 50% but after it is flipped it either certainly landed heads or it certainly did not.

This paper presents a quantified temporal probability logic that can represent all of these aspects of time, chance, and action. The model represents time in terms of possible world-histories and represents and chance by defining probabilities over the world-histories. Actions are represented as events. The logical language integrates both modal and probabilistic constructs and thus can represent and distinguish between possibility, probability, and truth. For example, the language allows us to write sentences that

- 1) describe concurrent actions:

- It is not possible for me to both raise and lower my arm at the same time.

- 2) describe conditions during an action that influence the probabilistic effects of the action:

- Holding the oven temperature constant increases the likelihood that the souffle will turn out right.

3) describe conditions not influenced by an action:

- Nobody can influence the chance of rain by clapping his hands.

4) mix statements about probability and inevitability:

- There is a 50% chance that by noon the train crash will be inevitable.

5) and distinguish between truth and probability:

- I won the lottery even though it was unlikely.

Numerous researchers have developed temporal logics for reasoning about plans and actions [McDermott, 1982, Allen, 1984, Haas, 1985, Pelavin and Allen, 1986, Shoham, 1987]. Others have developed logics of probability [Fagin *et al.*, 1988, Bacchus, 1988, Halpern, 1991]. But no work has addressed all these issues in a comprehensive logical framework. Such a framework is necessary for representing and reasoning about actions and plans in uncertain domains.

The theory of computing literature contains several examples of logics that can represent both time and probability [Lehmann and Shelah, 1982, Hart and Shair, 1984, Halpern and Tuttle, 1989]. The focus of these logics is on reasoning about probabilistic programs and distributed systems. The logics do not attempt to model effects or to distinguish between different types of temporal objects such as facts and events; hence, they are not suitable for reasoning about actions and plans.

The logic presented here extends the capabilities of current planning logics by providing a probabilistic treatment of effects, concurrent actions, and conditions that hold or change during execution of an action. The logic combines aspects of Pelavin's [1988] temporal planning logic, van Fraassen's [1980] models of objective chance, Fagin, Halpern, and Megiddo's [1988] probability logic, and Haddawy and Frisch's [1990] staged probability models. The present work builds on the author's previous work [Haddawy, 1990] by adding a theory of quantification and a more refined representation of events.

This paper does not address some traditional planning issues such as qualification and persistence. However, the developed logic does provide a framework in which one can explore these traditional planning issues for domains in which chance is a factor. The logic is intended to be used as a tool in the representation of a new class of interesting and difficult planning problems. It will enable us to design planning algorithms for these problems and to prove the algorithms' correctness.

The remainder of the paper is organized as follows. Section 2 presents the ontology of the logic. Section 3 describes the syntax of the logical language and shows

how the example sentences above can be represented. Section 4 presents the model theory and the semantic constraints necessary to achieve the desired intuitive properties of the logic. Section 5 discusses axiomatizability and presents a set of useful properties. Section 6 discusses the relation between probabilities over time. Section 7 discusses the representation of necessary properties of effects. Section 8 presents two examples that illustrate the use of the logic. Section 9 discusses related work and section 10 discusses directions for future research.

2 The Ontology

The present work focuses on modeling four aspects of the world: time, facts/events, possibility, and chance. Actions are taken to be events.¹

2.1 Time

Time is modeled as a collection of world-histories, each of which is one possible chronology or history of events throughout time. At any given point in time, some of the world-histories will naturally share a common past up to that time. Thus the world-histories form a tree structure that branches into the future. No special status is given to the time "now", so the temporal tree branches into the future relative to each point in time. The present work is only concerned with future-branching time because actions and other events can only affect the state of the world at times after their occurrence. At each point in time, the past is fixed – no occurrences in the world will cause it to change. But at each point in time the future might unfold in any number of ways, which are influenced by the agent's actions as well as by other events. So relative to any point in time, only one possible past exists, but numerous possible futures exist.²

2.2 Facts & Events

The world is described in terms of facts and events. Facts tend to hold and events tend to occur over intervals of time. Each fact is associated with the set of temporal intervals over which it holds. If a fact holds over an interval then it holds over all subintervals of that interval. So, for example, if my car is red over a period of time, then it is red throughout that time. Events are somewhat more complex than facts. First,

¹Actions actually differ from other events in that they are directly brought about by an agent. Haddawy [1991] presents a more refined representation that captures this distinction

²For a more thorough discussion of why a future-branching time model is appropriate for representing effects see the article by Mellor [1987].

one must distinguish between *event types* and *event tokens*. An event type is a general class of events and an event token is a specific instance of an event type. For example, picking up a book is an event type, and picking up the blue book on my desk at 9:00am is an event token of that type. Event tokens are unique individuals. The interval over which an event token occurs is the unique interval containing the event token and an event token can occur at most once in any world-history. If I pick up a book during a time period, there is no smaller period of time during which the event of my picking up the book can be said to have occurred. On the other hand, numerous tokens of a given event type may occur during a particular interval. For example, I can pick up one book with my right hand and one with my left hand concurrently. So if a token of an event type occurs over an interval, it is possible for another token of that type to occur over a subinterval, but it is not necessary as it is in the case of facts. This paper deals with event types, which for brevity are simply referred to as events.

The fact/event dichotomy just described is a simplification of the true situation. As Shoham [1987] has shown, there are many different types of facts and events, characterized by their temporal properties. Although Shoham's refined categories of fact types constitute a more useful and accurate picture of the world than the simple fact/event dichotomy, the fact/event categorisation will be used for simplicity of exposition. Extending the work to encompass Shoham's categories is completely straightforward.

2.3 Possibility

Something is possible if it either was true in the past or could be true in the future. So possibility is defined relative to the tree of possible futures. Because the present work is concerned with representing action effects, possibility is taken relative to a given point in time, with respect to which the past is inevitably true or inevitably false. In this way, actions and other events can only affect the possibility of future facts and events.

2.4 Chance

Chance is introduced by defining probabilities over the tree of possible futures. Like possibility, probability is taken relative to a given point in time, with respect to which the probability of the past is either zero or one. In this way, actions and other events can only affect the probabilities of future facts and events. This type of probability is objective, as opposed to subjective. Subjectively the past can be uncertain but objectively it is completely determined. For example, subjectively I may be uncertain as to whether the train has left or

not, yet objectively it either certainly left or it certainly did not and, furthermore, there is nothing I can do now to change that. The other property imposed on objective probability is that the probability be completely determined by the history up to the current time. So objective probability is purely a function of the state of the world. In contrast, subjective probability is a function not only of the state of the world but also of the epistemic state of an agent.

The present characterization of objective probability is not to be confused with the frequentist interpretation of probability [Venn, 1866, von Mises, 1957] which is also often called objective probability. Our model of chance is motivated by the subjectivist theories of objective probability [Lewis, 1980, Skyrms, 1980b, van Fraassen, 1980], which define it in terms of properties that one would expect a rational agent to believe objective probability to possess. Although the current work does not include subjective probabilities, the model is completely consistent with the subjectivist framework.

3 Syntax

To formalize the concepts discussed in the ontology, we now define the language of temporal probability logic \mathcal{L}_{tp} . To refer to facts and event types occurring in time the language contains two predicates. $HOLDS(FA, t_1, t_2)$ is true if fact FA holds over the interval t_1 to t_2 , and $OCC(EV, t_1, t_2)$ is true if event EV occurs during the interval t_1 to t_2 . The language contains three modal operators to express inevitability, possibility, and chance. We write $\Box_t(\phi)$ to indicate that ϕ is inevitably true at time t and $\Diamond_t(\phi)$ to say that ϕ is possibly true at time t . We write $P_t(\phi) \geq \alpha$ to say that the probability of ϕ at time t is at least α . The sentence form $P_t(\phi) \leq \alpha$ is used as a shorthand for $P_t(\neg\phi) \geq 1 - \alpha$. Similarly $=$, $>$, and $<$ are used as shorthand for their corresponding sentences. Following the syntax of Fagin, Halpern, and Megiddo's [1988] probability logic of polynomial weight formulas, polynomial combinations of probability operators will be allowed. Thus the language can express sentences like " ϕ is at least twice as likely as ψ ": $P_t(\phi) \geq 2P_t(\psi)$. This is particularly useful for writing sentences about conditional probability. The probability of ϕ given ψ is traditionally defined as

$$\text{prob}(\phi|\psi) = \text{prob}(\phi \wedge \psi)/\text{prob}(\psi).$$

If the probability of the conditioning sentence ψ is zero, then the conditional probability is undefined. In this case, a conditional probability sentence like $\text{prob}(\phi|\psi) = \alpha$ can be assigned neither the value true nor the value false. Rather than introducing a new conditional probability operator and dealing with this truth assignment problem, sentences about conditional probability can simply be written in the form

$$P_t(\phi \wedge \psi) \geq \alpha P_t(\psi).$$

1. If $tterm_1, tterm_2$ are temporal terms then $tterm_1 = tterm_2$, $tterm_1 \preceq tterm_2$, and $tterm_1 \prec tterm_2$ are wffs.
2. If $tterm_1, tterm_2$ are temporal terms, $nterm_1, \dots, nterm_n$ are nontemporal terms, and rf is an n-ary fact relation symbol then $HOLDS(rf(nterm_1, \dots, nterm_n), tterm_1, tterm_2)$ is a wff.
3. If $tterm_1, tterm_2$ are temporal terms, $nterm_1, \dots, nterm_n$ are nontemporal terms, and re is an n-ary event relation symbol then $OCC(re(nterm_1, \dots, nterm_n), tterm_1, tterm_2)$ is a wff.
4. If ϕ_1 and ϕ_2 are wffs then so are $\neg\phi_1$, $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, and $\phi_1 \rightarrow \phi_2$.
5. If ϕ is a wff and $z \in TV \cup V$ then $\forall z\phi$ and $\exists z\phi$ are wff's.
6. If ϕ is a wff and $tterm$ is a temporal term then $\Box_{tterm}(\phi)$ and $\Diamond_{tterm}(\phi)$ are wffs.
7. If ϕ_1, \dots, ϕ_k are wffs, $\theta_0, \dots, \theta_k$ are real numbers, and $tterm$ is a temporal term then $\theta_0 P_{tterm}(\phi_1) \cdot \dots \cdot P_{tterm}(\phi_k) + \theta_1 P_{tterm}(\phi_1) \cdot \dots \cdot P_{tterm}(\phi_{k-1}) + \dots + \theta_k \geq 0$ is a wff.

Figure 1: Definition of the well-formed formulas

Note that this sentence is true for all values of α if $prob(\psi) = 0$. The standard conditional probability notation will be used to syntactically denote a sentence of the above form:

$$P_t(\phi|\psi) \geq \alpha.$$

The probability operator can be arbitrarily nested and combined with the inevitability operator, allowing us to write complex sentences like:

$$P_{t_1}(\Box_{t_2}\phi \wedge P_{t_3}(\psi = \alpha) = \beta.$$

The lexicon of the language consists of the following disjoint sets of nonlogical symbols:³ TC , a set of time point symbols; C , a set of constant symbols; TV , a set of temporal variables; V , a set of variables; $TFCT$, a set of temporal function symbols; FCT , a set of function symbols; RF , a set of fact relation symbols; and RE , a set of event relation symbols.

The set of *temporal terms* is composed of all members of TC , all members of TV , as well as all symbols of the form $f(tterm_1, \dots, tterm_n)$, where $f \in TFCT$ and $tterm_1, \dots, tterm_n$ are temporal terms. The set of *nontemporal terms* is defined similarly.

The set of well-formed formulas combining the logical and nonlogical symbols is recursively defined as shown in figure 1.

The five sentences from the introduction can be represented as follows.

- It is not possible for me to raise and lower my arm at the same time.
 $\forall t_0, t_1, t_2 \neg \Diamond_{t_0}[OCC(\text{raise}(\text{arm}), t_1, t_2) \wedge OCC(\text{lower}(\text{arm}), t_1, t_2)]$
- Holding the oven temperature constant increases the likelihood that the souffle will turn out right.
 $(\text{now} \preceq t_1 \preceq t_2) \wedge P_{\text{now}}(HOLDS(\text{done-right}(\text{sfl}), t_2, t_2))$

³The definitions of the lexicon and the temporal and nontemporal terms follow the syntax of Shoham [1987].

$$\begin{aligned} & OCC(\text{bake}(\text{sfl}), t_1, t_2) \wedge \\ & HOLDS(\text{const}(\text{temp}(\text{oven})), t_1, t_2)) > \\ & P_{\text{now}}(HOLDS(\text{done-right}(\text{sfl}), t_2, t_2) | \\ & OCC(\text{bake}(\text{sfl}), t_1, t_2)) \end{aligned}$$

- Nobody can influence the chance of rain by clapping his hands.
 $\forall t_1, t_2, t_3, t_4, x$
 $P_{\text{now}}(OCC(\text{rain}, t_1, t_2) | OCC(\text{clap}(x, \text{hands}(x)), t_3, t_4)) = P_{\text{now}}(OCC(\text{rain}, t_1, t_2))$
- There is a 50% chance that by noon the train crash will be inevitable.
 $P_{\text{now}}(\exists t_1 (t_1 \preceq \text{noon}) \wedge \Box_{t_1} OCC(\text{crash}(\text{train}), t_2, t_3)) = .5$
- I won the lottery even though it was unlikely
 $(t_0 \prec t_1 \prec t_2 \prec \text{now}) \wedge P_{t_0}(OCC(\text{win-lottery}, t_1, t_2)) = .0001 \wedge OCC(\text{win-lottery}, t_1, t_2)$

4 Semantics

A model is a tuple $\langle W, T, \leq, D, TFN, FN, RLF, RLE, FA, EVENTS, EV, R, X, PR, F \rangle$, where:

- W is the set of possible world-histories.
- T is a totally ordered set of time points, corresponding to the reals.
- \leq is a total ordering on T .
- D is the non-empty domain of individuals, disjoint from T .
- TFN is the set of temporal functions: $T^k \rightarrow T$.
- FN is the set of nontemporal functions: $D^k \rightarrow D$.
- RLF is the set of fact relations: $D^k \rightarrow 2^{(T \times T) \times W}$.
- RLE is the set of event relations: $D^k \rightarrow 2^{(T \times T) \times W}$.
- FA is the set of facts, a subset of $2^{(T \times T) \times W}$.
- $EVENTS$ is the set of event tokens, a subset of $2^{(T \times T) \times W}$.

- EV is the set of event types, a subset of 2^{EVENTS} .
- R is an accessibility relation defined on $T \times W \times W$. $R(t, w_1, w_2)$ means that world-histories w_1 and w_2 are indistinguishable up to time t . The set of all world-histories R accessible from w at time t will be designated R_t^w . For each time t , the R_t^w partition the world-histories into sets of equivalence classes indistinguishable up to t .
- \mathcal{X} is a σ -algebra over W , containing all the sets corresponding to wff's in the language, as well as all R -equivalence classes of world-histories.
- PR is a probability assignment function that assigns to each time $t \in T$ and world-history $w \in W$ a probability distribution μ_t^w defined over \mathcal{X} .
- F is the denotation function, defined as follows:
 $TC \rightarrow T, C \rightarrow D, TFCT \rightarrow TFN, FCT \rightarrow FN, RF \rightarrow RLF, RE \rightarrow RLE$.⁴

In Section 2 the ontology of the logic was discussed from an intuitive standpoint. In order to obtain the desired intuitive properties, a number of constraints must be imposed on the models. These constraints, labeled (C1)–(C7) are presented in the following discussion.

The future branching temporal tree is defined in terms of the R relation over world-histories. To capture the property that time does not branch into the past, we say that if two world-histories are indistinguishable up to time t_2 then they are indistinguishable up to any earlier time:

(C1) If $t_1 \leq t_2$ and $R(t_2, w_1, w_2)$ then $R(t_1, w_1, w_2)$.

Since R just represents the indistinguishability of histories up to a time t , for a fixed time R is an equivalence relation.

(C2) For a fixed time R is reflexive, transitive, and symmetric.

Figure 2 illustrates how the R relation ties together the different world-histories to form the temporal tree structure.

Facts and events hold and occur in various world-histories at various times. Thus, we identify facts and events with sets of \langle temporal interval, world \rangle pairs. Following Shoham [1987], we take a temporal

⁴The reader will note that constants, functions, and relations are rigid with respect to both time and world-history. Rigidity with respect to world-history simplifies the logic somewhat and could be relaxed without much effort. Non-rigidity with respect to time seems to be basically incompatible with the current framework since it would result in inconsistencies like the following. At time t_1 a term like *Blue(car)* could denote the fact that my car is blue from t_0 to t_3 and at time t_2 it could denote the fact that my car is blue from t_4 to t_6 .

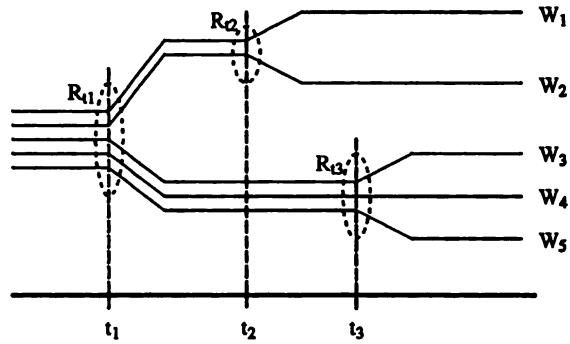


Figure 2: Structure imposed by the R accessibility relation.

interval to be simply an ordered pair of time points. So a fact(event) is a set of elements of the form $\langle\langle t_1, t_2\rangle, w\rangle$, where $t_1, t_2 \in T$, $t_1 \leq t_2$, and $w \in W$. If $\langle\langle t_1, t_2\rangle, w\rangle \in A$ then fact(event) A holds(occurs) during interval $\langle t_1, t_2\rangle$ in world-history w .

As mentioned earlier, facts and events differ in their temporal properties. This distinction is captured by the following two semantic constraints. If a fact holds over an interval, it holds over all subintervals:

(C3) If $t_1 \leq t_2 \leq t_3 \leq t_4$, $t_1 \neq t_3$, $t_2 \neq t_4$, $fa \in FA$ and $\langle\langle t_1, t_4\rangle, w\rangle \in fa$ then $\langle\langle t_2, t_3\rangle, w\rangle \in fa$.

An event token occurs only once in each world-history:

(C4) If $evt \in EVENTS$, $\langle\langle t_1, t_2\rangle, w\rangle \in evt$, and $\langle\langle t_3, t_4\rangle, w\rangle \in evt$ then $t_1 = t_3$ and $t_2 = t_4$.

If two worlds are indistinguishable up to a time then they must share a common past up to that time. And if they share a common past up to a given time, they must agree on all facts and events up to that time. To enforce this relationship, we impose the constraint that if two world-histories are R related at time t , they must agree on all facts(events) that hold(occur) over intervals ending before or at the same time as t :

(C5) If $t_1 \leq t_2$ and $R(t_2, w_1, w_2)$ then $\langle\langle t_0, t_1\rangle, w_1\rangle \in A$ iff $\langle\langle t_0, t_1\rangle, w_2\rangle \in A$, where A is a fact or event.

Section 2 mentions two desired characteristics of the probability operator. The first is that the probability at a time t be completely determined by the history up to that time. This is achieved by the following constraint.

(C6) For all $X \in \mathcal{X}, t \leq t'$, and w, w' such that $R(t, w, w')$,
 $\mu_t^w(R_t^{w'}) > 0 \rightarrow \mu_{t'}^w(X) = \mu_{t'}^w(X|R_t^{w'})$.

If history determines chance, then two worlds that share a common history up to a time should have

1. If u is a variable then $[u]^{M,w,s} = g(u)$.
2. If α is a non-logical constant then $[\alpha]^{M,w,s} = F(\alpha)$.
3. If $r = f(trm_1, \dots, trm_n)$ is a term (temporal or non-temporal) then $[r]^{M,w,s} = [f]^{M,w,s}([trm_1]^{M,w,s}, \dots, [trm_n]^{M,w,s})$.
4. $[trm_1 \preceq trm_2]^{M,w,s} = \text{true iff } [trm_1]^{M,w,s} \leq [trm_2]^{M,w,s}$.
5. $[trm_1 = trm_2]^{M,w,s} = \text{true iff } [trm_1]^{M,w,s} = [trm_2]^{M,w,s}$.
6. $[HOLDS(rf(nirm_1, \dots, nirm_n), trm_1, trm_2)]^{M,w,s} = \text{true iff } \langle ([trm_1]^{M,w,s}, [trm_2]^{M,w,s}), w \rangle \in F(rf)([nirm_1]^{M,w,s}, \dots, [nirm_n]^{M,w,s})$.
7. $[OCC(re(nirm_1, \dots, nirm_n), trm_1, trm_2)]^{M,w,s} = \text{true iff } \langle ([trm_1]^{M,w,s}, [trm_2]^{M,w,s}), w \rangle \in e$ for some $e \in F(re)([nirm_1]^{M,w,s}, \dots, [nirm_n]^{M,w,s})$.
8. $[\neg\phi]^{M,w,s} = \text{true iff } [\phi]^{M,w,s} \neq \text{true}$.
9. $[\phi_1 \wedge \phi_2]^{M,w,s} = \text{true iff } [\phi_1]^{M,w,s} = \text{true and } [\phi_2]^{M,w,s} = \text{true}$.
10. $[\forall z \phi]^{M,w,s} = \text{true iff } [\phi]^{M,w,s[d/z]} = \text{true for all } d \in D$ if $z \in V$ and for all $d \in T$ if $z \in TV$. The expression $g[d/z]$ denotes the assignment of values to variables that is identical to assignment g with the possible exception that element d is assigned to variable z .
11. $[\Box_{trm}(\phi)]^{M,w,s} = \text{true iff } [\phi]^{M,w',s} = \text{true for every } w'$ such that $R([trm]^{M,w,s}, w, w')$.
12. $[\mathbb{P}_{trm}(\phi) \geq 0]^{M,w,s} = \text{true iff } \mu_{[trm]^{M,w,s}}^w(\{w' \in R_{[trm]^{M,w,s}}^w \mid [\phi]^{M,w',s} = \text{true}\}) \geq 0$.

Figure 3: Semantic definitions

identical probability distributions at that time. This follows from constraint (C6): For all w, w' such that $R(t, w, w')$

$$\mu_t^w(X) = \mu_t^w(X | R_t^{w'}) = \mu_t^w(X)$$

The second desired characteristic is that the probability of a past fact or event should be either zero or one, depending on whether or not it actually happened. This follows from (C6) together with the constraint,

$$(C7) \quad \mu_t^w(R_t^w) > 0.$$

To show that the probability of the past is zero or one we simply instantiate X in (C6) to the equivalence class R_t^w :

$$\mu_t^w(R_t^w) = \mu_t^w(R_t^w | R_t^w) = 1.$$

Defining the probabilities in this way makes good intuitive sense if we look at the meaning of R . R_t^w designates a set of world-histories that are objectively possible with respect to w at time t . It is natural that the set of world-histories that are objectively likely with respect to w at time t should be a subset of the ones that are possible.

Given the models described above, the semantic definitions for the well-formed formulas can now be defined. Denotations are assigned to expressions relative to a model, a world-history within the model, and an assignment of individuals in the domain to variables. The denotation of an expression ϕ relative to a model

M and a world-history w , and a variable assignment g is designated by $[\phi]^{M,w,s}$. The variable assignment function g maps each temporal variable to a time point and each nontemporal variable to a nontemporal object. The semantic definitions for the well-formed formulas are shown in figure 3.

The logical operators \vee , \rightarrow , and \exists are defined in terms of \neg , \wedge , and \forall in the usual way. Possibility \Diamond is defined in terms of inevitability as $\Diamond_t(\phi) \equiv \neg \Box_t(\neg\phi)$. The interesting definitions are the last two. Definition 11 says that a sentence is inevitable in a world w at a time t iff it is true in all worlds indistinguishable from w up to time t . Definition 12 says that the probability of a sentence ϕ is at least α in a world w at a time t iff the probability of those accessible worlds in which ϕ is true is at least α .

A sentence ϕ is *satisfied* by a model M at a world w if it is assigned the value true in that model and world. A sentence is *valid* if it is satisfied by every model at every world.

5 Properties

Fagin, Halpern, and Megiddo [1988] present a propositional probability logic of polynomial weight formulas. That logic corresponds to a propositional version of the probabilistic component of L_{tp} . They show that a complete axiomatisation of their logic requires the theory of real closed fields. This theory can only be

expressed in a language that allows quantification over probability values. Since \mathcal{L}_{tp} does not allow such quantification, it can not be completely axiomatized. But what if we add quantifiers over probability values?

Halpern [1991] describes a first-order probability logic \mathcal{L}_2 that allows quantification over real probability values. Probability in \mathcal{L}_2 is defined in terms of a single distribution over possible worlds. If we add quantification over probability values to \mathcal{L}_{tp} , then Halpern's \mathcal{L}_2 becomes a special case of the resulting logic. \mathcal{L}_2 corresponds roughly to the case in which the temporal probability models have only a fixed time point and a single equivalence class of worlds at that time point, and the sentences contain no modal operators. Halpern shows that the logic \mathcal{L}_2 is not recursively enumerable and hence is not completely axiomatizable. It follows that the extended version of \mathcal{L}_{tp} would not be axiomatizable either.

The lack of existence of a complete axiomatization for \mathcal{L}_{tp} is not seen as a serious drawback for two reasons. First, the logic is intended to be used in the design and analysis of planning algorithms that are sound with respect to the logic and possibly complete with respect to a subset of the logic. In fact, the author knows of no planning algorithms that are complete even for first-order logic.

Second, sound axioms and rules of inference can be provided that allow useful inferences to be made. The following properties expressed by valid sentence schemas allow many useful inferences.⁵

(P1) Facts hold over their subintervals:

$$\forall(t_1 \preceq t_2 \preceq t_3 \preceq t_4) \\ \text{HOLDS}(\phi, t_1, t_4) \rightarrow \text{HOLDS}(\phi, t_2, t_3)$$

(P2) Valid sentences are inevitable:

$$\text{If } \phi \text{ is valid then } \square_t \phi$$

(P3) The past is inevitable. For example,

$$\forall(t_0 \preceq t_1 \preceq t_2) [\square_{t_2}(\text{HOLDS}(\phi, t_0, t_1)) \vee \\ \square_{t_2}(\neg \text{HOLDS}(\phi, t_0, t_1))]$$

$$\forall(t_1 \preceq t_2) \diamondsuit_{t_2} [P_{t_2}(\phi) = \alpha] \rightarrow \square_{t_2} [P_{t_2}(\phi) = \alpha]$$

(P4) Inevitability (and hence impossibility) persists:

$$\forall(t_1 \preceq t_2) [\square_{t_1} \phi \rightarrow \square_{t_2} \phi]$$

Certainty does not persist since semantic constraint (C6) only applies to equivalence classes of worlds of positive measure.

⁵A rich set of axioms and inference rules is presented in [Haddawy, 1991].

(P5) If something is inevitable, it is certain:

$$\square_t(\phi) \rightarrow P_t(\phi) = 1$$

The converse is not valid since in an uncountably infinite space, a set of worlds can be possible yet still have measure zero.

(P6) Probability is non-negative:

$$P_t(\phi) \geq 0$$

(P7) Probability is additive:

$$P_t(\phi) = P_t(\phi \wedge \psi) + P_t(\phi \wedge \neg\psi)$$

Several further properties now follow directly from these seven. Properties (P6) and (P7) are two of the well-known three axioms of probability. The third axiom is a consequence of properties (P2) and (P5):

(P8) Valid sentences are certain:

$$\text{If } \phi \text{ is valid then } P_t(\phi) = 1$$

From (P6), (P7), and (P8) it follows that

(P9) Stronger sentences have lower probability:

$$(\text{For any } \phi, \psi | \phi \models \psi) P_t(\psi) \geq P_t(\phi)$$

Properties (P1) and (P9) entail that

(P10) Facts have higher probability of holding over their subintervals:

$$\forall(t_1 \preceq t_2 \preceq t_3 \preceq t_4) \\ P_t(\text{HOLDS}(\phi, t_2, t_3)) \geq P_t(\text{HOLDS}(\phi, t_1, t_4))$$

From properties (P3) and (P5) we have that

(P11) The past is certain:

$$\forall(t_0 \preceq t_1 \preceq t_2) [P_{t_2}(\text{HOLDS}(\phi, t_0, t_1)) = 1 \vee \\ P_{t_2}(\text{HOLDS}(\phi, t_0, t_1)) = 0]$$

6 Miller's Principle

As a consequence of the characteristics we have imposed on probability, an interesting relationship holds between probabilities assigned to the same sentence at various times. The probability of a sentence at some time given that its probability at some future time is at least α should be at least α : $\forall(t_1 \preceq t_2) P_{t_1}(\phi | P_{t_2}(\phi) \geq \alpha) \geq \alpha$. This relation is called *Miller's principle* and several nontemporal variants of it were first suggested by Brian Skyrms [1980a] as possible constraints on higher-order probabilities. An informal intuitive argument that Miller's principle holds in this logic is presented here. The formal proof is given in the appendix.

By constraint (C6), temporally qualified probabilities can be written as conditional probabilities: $\mu_{t_2}(A) = \mu_{t_1}(A | \text{history up to } t_2)$. Then what should be the chance of A at time t_1 given that events unfold to time t_2 , $t_1 \leq t_2$, such that the chance of A at t_2 is α ? At time t_1 , as far as events that determine the chance of A are concerned, all possible futures up to time t_2 are equivalent. So the probability should just be α :

$$\begin{aligned}\mu_{t_1}(A | \mu_{t_2}(A) = \alpha) &= \\ \mu_{t_1}(A | \mu_{t_1}(A | \text{history up to } t_2) = \alpha) &= \\ \mu_{t_1}(A | \text{hist up to } t_2 \text{ is such that } \mu_{t_2}(A) = \alpha) &= \alpha\end{aligned}$$

In the object language, Miller's principle can be stated as the sentence schema:

$$\forall(t \preceq t') P_t(\phi \wedge P_{t'}(\phi) \geq \alpha) \geq \alpha \cdot P_t(P_{t'}(\phi) \geq \alpha)$$

Every instance of this schema is valid in the temporal probability models.

Miller's principle is useful for two reasons. First, it formalizes our intuitions about the relation between chance at various times. For example, suppose that I have two coins, one fair coin and one with a 70% chance of heads and that I am going to choose one coin and flip it. What is the chance now of heads given that I will choose the biased coin? By intuition and Miller's principle it is 70%. Notice that as Skyrms [1980a, Appendix 2] has pointed out, if we have a sentence in the language corresponding to the-biased-coin-is-chosen and appropriate conditional probabilities we can always represent such higher-order probability statements with simple probabilities. But we may not always have such sentences readily available.

The second useful consequence of Miller's principle is that it allows the current chance of facts and events to be inferred from the chances of their future chances: the probability at a given time is the expected value of the probability at any future time. This will be called the *expected value principle*. Suppose I am going to choose at random between the two coins above. There is a 50% chance that the chosen coin will have a 70% chance of landing heads and a 50% chance that the coin will have a 50% chance of landing heads. By the Miller's principle, it follows that there is now a 60% chance that the coin flip will result in heads.

7 Representing Effects of Actions

It was stated in the introduction that one of the intended uses of this logic is for reasoning about action effects. We now show how some natural properties of effects follow directly from the models of chance and possibility. We describe positive effects, i.e. conditions that actions bring about, but the discussion holds for negative effects as well. In describing effects we must distinguish between two cases: effects whose chance is less than one and effects whose chance equals one. The reason this distinction is necessary is that there may

be a condition that has probability one of occurring yet is not necessary. If an action made the condition necessary, the condition would be considered a positive effect of the action.

If the chance of EFF is less than one then three conditions are necessary for EFF to be a positive effect of an act A : i) EFF cannot temporally precede A ; ii) A must have a positive chance of occurring; and iii) A must positively influence the probability of EFF .⁶ If the chance of EFF is equal to one then four conditions are necessary for EFF to be an effect of an act A : i) EFF cannot temporally precede A ; ii) EFF cannot be necessarily true; iii) A must be possible and; iv) A must necessarily bring about EFF .

Suppose act A occurs in the interval t_A to $t_{A'}$ and let EFF be a fact or event: $HOLDS(FACT, t_E, t'_E)$ or $OCC(EVENT, t_E, t'_E)$. Then the above conditions can be stated in the logic as

1. $P_{t_A}(EFF) < 1$ and
2. Temporal non-succession: $t_A \prec t'_B$
3. Positive chance of action: $P_{t_A}(OCC(A, t_A, t'_A)) > 0$
4. Positive influence: $P_{t_A}(EFF | OCC(A, t_A, t'_A)) > P_{t_A}(EFF)$
or
5. $P_{t_A}(EFF) = 1$ and
6. Temporal non-succession: $t_A \prec t'_B$
7. Non-necessity of effect: $\neg \Box_{t_A} EFF$
8. Possibility of action: $\Diamond_{t_A} OCC(A, t_A, t'_A)$
9. Necessity of influence: $\Box_{t_A}(OCC(A, t_A, t'_A) \rightarrow EFF)$

Due to the way the temporal probability models have been defined, some of the above conditions entail others. It follows from the definition of objective probability that condition 4) entails both conditions 2) and 3). If condition 4) is expanded out into its proper form in the logic it becomes.

$$\begin{aligned}P_{t_A}(EFF \wedge OCC(A, t_A, t'_A)) &> \\ P_{t_A}(EFF) \cdot P_{t_A}(OCC(A, t_A, t'_A)) &\end{aligned}$$

It is clear that if $P_{t_A}(OCC(A, t_A, t'_A)) = 0$ then the sentence is false. So if 4) holds, 3) must hold. Next, if 1) is false then $t'_B \preceq t_A$. Since the probability of past events is either zero or one,

$$\text{either } P_{t_A}(EFF) = 0 \text{ or } P_{t_A}(EFF) = 1.$$

Either case contradicts 4). So if 4) holds, 2) must also hold.

⁶Note that these conditions are similar to Suppes [1970] *prima facie causality conditions*.

Next we show that the temporal non-succession condition 6) follows from conditions 7), 8) and 9). Conditions 8) and 9) together entail that

$$\diamond_{t_4} \text{EFF}$$

which by definition is equivalent to

$$\neg \square_{t_4} \neg \text{EFF}.$$

Since the past is inevitable, this along with condition 7) entails condition 6).

This result shows that the models have captured the temporal relation between actions and effects as intended - actions cannot affect the past. As a consequence of this result, if we use conditions 3), 7), 8), and 9) to define what is necessary for a plan to achieve a goal then we can prove that actions after the time of the goal cannot contribute to achieving the goal.

8 Examples

In this section two examples illustrating the use of the logic are presented. The first example shows how the logic can be used to distinguish between factors that an agent cannot influence because they are inevitable and factors the agent cannot influence because they are simply outside the control of his available actions. Consider a train driving along some tracks. The tracks are analogous to the temporal tree - the train travels in only one direction and at certain points in time the tracks branch. Some of the tracks on which the train can run have another train coming in the opposite direction and if the train is on one of these, a crash will occur. We have an action switch-safe that will switch the train to a safe track if one is available. Suppose that at time t_1 there is a 50% chance that the train will be switched to a set of tracks where a crash is inevitable. Suppose further that there is a 25% chance that at t_1 the train will get switched to a set of tracks where a crash is not certain but where the switch-safe action has no effect. What is the chance that we can influence the chance of a crash through the switch-safe action? The situation can be described by the following two sentences:

$$P_{\text{now}}(\square_{t_1} \text{OCC(crash, } t_2, t_3)) = .5 \quad (1)$$

$$\begin{aligned} P_{\text{now}}(P_{t_1}(\text{OCC(crash, } t_2, t_3)) < 1 \wedge \\ P_{t_1}(\text{OCC(crash, } t_2, t_3) | \text{OCC(ss, } t_s, t'_s)) = \\ P_{t_1}(\text{OCC(crash, } t_2, t_3)) = .25 \end{aligned} \quad (2)$$

One possible model for these sentences is shown in figure 4.

By the properties that inevitability persists (P4) and stronger sentences have lower probability (P9), it follows from sentence (1) that

$$P_{\text{now}}(\square_{t_1} \text{OCC(crash, } t_2, t_3)) \geq .5 \quad (3)$$

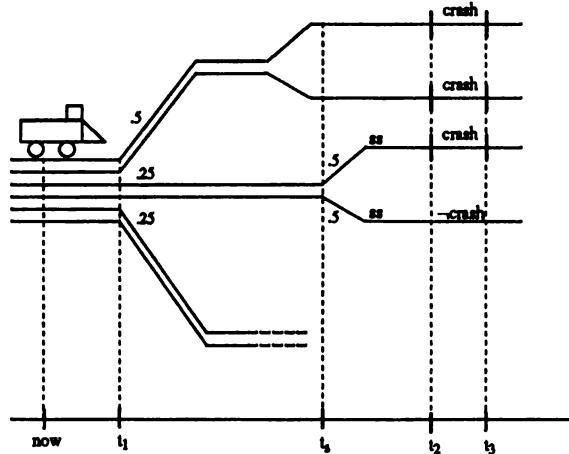


Figure 4: Possible model for train example.

By the relation between inevitability and probability (P5) and the axioms of probability (P6-P8), it follows from sentence (3) that

$$\begin{aligned} P_{\text{now}}(P_{t_1}(\text{OCC(crash, } t_2, t_3) | \text{OCC(ss, } t_s, t'_s)) = \\ P_{t_1}(\text{OCC(crash, } t_2, t_3)) \geq .5 \end{aligned} \quad (4)$$

Finally since $\square_{t_1} \text{OCC(crash, } t_2, t_3)$ and $P_{t_1}(\text{OCC(crash, } t_2, t_3)) < 1$ are mutually exclusive, by property (P7) sentences (2) and (4) can be combined to yield

$$\begin{aligned} P_{\text{now}}(P_{t_1}(\text{OCC(crash, } t_2, t_3) | \text{OCC(ss, } t_s, t'_s)) = \\ P_{t_1}(\text{OCC(crash, } t_2, t_3)) \geq .75 \end{aligned} \quad (5)$$

So there is at most a 25% chance that we can influence the chance of a crash by executing the switch-safe action.

The next example is a modified version of an example presented by Pelavin [1988]. It illustrates how the ability of the logic to represent both probability and possibility can be used to reason about the chance that two actions can co-occur. Suppose that I am going shopping this evening and want to carry both grocery bags to the car simultaneously. In most cases it is not possible to carry two bags if it is icy out. There is a 50% chance that it will be icy out this evening. What is the chance that carrying both bags simultaneously will not be a possible course of action? The situation can be described by the following three sentences:

$$\begin{aligned} \forall t, t' (\text{now} \preceq t \preceq t') \rightarrow \\ P_{\text{now}}(\neg \diamond_t [\text{OCC(carry(b1), } t, t') \wedge \\ \text{OCC(carry(b2), } t, t')]) | \\ \text{HOLDS(icy, } t, t')) = .8 \end{aligned} \quad (6)$$

$$P_{\text{now}}(\text{HOLDS(icy, } t_1, t_3)) = .5 \quad (7)$$

$$\text{now} \prec t_1 \prec t_2 \prec t_3 \quad (8)$$

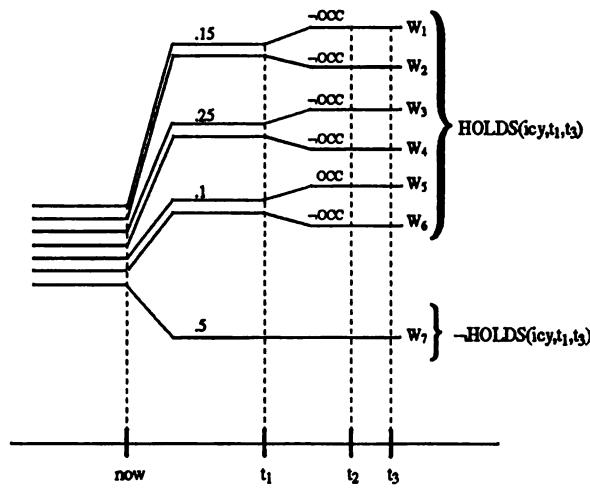


Figure 5: Possible model for carry example.

We would like to know the probability that

$$\neg \diamond_{t_1} [OCC(\text{carry}(b1), t_1, t_2) \wedge OCC(\text{carry}(b2), t_1, t_2)]$$

One possible model satisfying these sentences is shown in figure 5. The labels “OCC” and “ \neg OCC” designate the co-occurrence and non co-occurrence of the two actions, respectively. Note that in worlds $w_1 - w_4$ we have

$\neg \diamond_{t_1} [OCC(\text{carry}(b1), t_1, t_2) \wedge OCC(\text{carry}(b2), t_1, t_2)]$ and in w_5 and w_6 we have

$$\diamond_{t_1} [OCC(\text{carry}(b1), t_1, t_2) \wedge OCC(\text{carry}(b2), t_1, t_2)].$$

By the property that facts have higher probability of holding over their subintervals (P10), sentences (7) and (8) entail that

$$P_{\text{now}}(HOLD(S(\text{ic}, y, t_1, t_2))) \geq .5 \quad (9)$$

From sentences (6) and (9) it follows from property (P7) that

$$P_{\text{now}}(\neg \diamond_{t_1} [OCC(\text{carry}(b1), t_1, t_2) \wedge OCC(\text{carry}(b2), t_1, t_2)]) \geq .4$$

So there is at least a 40% chance that carrying both bags simultaneously will not be possible.

Furthermore, an upper bound on the current probability of the co-occurrence of the two actions can be calculated. By the relation between inevitability and probability (P5),

$$P_{\text{now}}(P_{t_1}(OCC(\text{carry}(b1), t_1, t_2)) \wedge OCC(\text{carry}(b2), t_1, t_2)) = 0 \geq .4$$

and by Miller's principle,

$$P_{\text{now}}(OCC(\text{carry}(b1), t_1, t_2) \wedge OCC(\text{carry}(b2), t_1, t_2)) \leq .6$$

9 Related Work

Pelavin develops a future-branching time logic for reasoning about planning problems involving concurrent actions and external events. He starts with Allen's [1984] linear temporal logic of time intervals and extends it with two modal operators, *INEV* and *IFTRIED*, to reason about future branching time and action effects, respectively. *INEV* is exactly our \square operator. *IFTRIED* is a counterfactual operator that associates the attempt of an action with the truth of a sentence. The semantics of the operator are based on Stalnaker's and Lewis's theories of counterfactuals. *IFTRIED* captures the temporal relation of action and effect—an action cannot affect the state of the world at any time preceding its attempt. Effects of general events cannot be represented in the logic.

The present work departs from Pelavin's framework in three major ways. First, points are taken to be the primitive temporal objects rather than intervals. Pelavin[1988, p84] himself notes that this results in a more natural definition of the accessibility relation. Second, the language can represent uncertainty. By representing uncertainty with objective probability, effects can be represented in terms of condition probability. Since probability can be conditioned on any sentence, the logic can represent effects of a wide variety of different phenomena. Counterfactuals as used by Pelavin are closely related to objective probability. Skyrms [1980a] provides an elegant probabilistic account of counterfactuals based on the notion of objective probability.

Third, Pelavin presents a more refined theory of actions than does the present paper. Pelavin distinguishes between actions and events as well as between the attempt of an action and its occurrence. These are important distinctions. Events may occur due to any number of causes but actions are directly brought about by an agent. An agent attempts an action and if the conditions are right, the action occurs. For example, I attempt to lift an object and if the object is not too heavy, I succeed in lifting it. The logic in this paper is extended in [Haddawy, 1991] to encompass these distinctions.

The temporal probability models we have presented are similar to those of van Fraassen[1980]. He presents a semantic theory that models subjective probability and objective chance, using a future-branching model of time points. In his models, objective probability can change with time but truth values cannot. He shows that a property equivalent to Miller's principle holds between subjective probability and objective chance, but not between objective chance at different times. He also does not provide a logical language.

Shoham [1989] presents a branching time logic that formalizes the relation between time, knowledge, and

action. Knowledge is defined in the standard way for modal logic and actions are defined as the ability to make choices among sets of world-histories. The model formalizes the notion that actions can only be performed under certain conditions. If we assume that only conditions up to the time of an action determine whether the action can be performed then Shoham's notion of action feasibility can be captured in \mathcal{L}_{tp} , by sentences of the form $\Diamond_{t_A} OCC(A, t_A, t'_A)$. Shoham's logic does not capture the fact that actions cannot influence the past.

10 Future Research

This paper has presented a logic for reasoning about objective probability. It is important for a planning representation to be able to represent the state of knowledge of the planning agent, e.g. to reason about actions that provide information. This can be done by introducing subjective probabilities into the logic. Several philosophers have discussed the problem of combining subjective probability and objective chance [Skyrms, 1980a, Appendix 2][Lewis, 1980, van Fraassen, 1980]. The general consensus is that agents have subjective beliefs concerning objective chance and the two are related by certain constraints, although there is disagreement as to precisely what the constraints should be. It can be shown that such a hybrid representation of beliefs is necessary in order to make rational decisions in some cases where causality is a factor [Skyrms, 1980a, ch IIC][Lewis, 1981, Maher, 1987]. Subjective probabilities are consistent with the present model theory. They can be modeled by defining probability functions over all worlds, not just the accessible ones. But the addition of subjective probabilities raises questions concerning the relation of belief to actions. These questions are addressed by Moore [1985] and Haas[1985] in a non-probabilistic setting.

Previous work [Halpern, 1989, Haddawy and Frisch, 1990] has shown that probability logic can be viewed as a generalization of modal logic. It has been shown [Gaifman, 1988, Haddawy and Frisch, 1990] that the logics corresponding to staged probability models, similar to the temporal probability models presented here, are closely related to certain modal logics. It would be interesting to see whether the probability logic presented in this paper corresponds to some temporal modal logic. Such a modal logic would be useful in providing a qualitative representation of probabilistic information.

Appendix: Proof of Miller's Principle

We first prove the expected value property and then use it to prove Miller's principle. Let t, t' be two time points $t \leq t'$ and consider the R-equivalence classes of

worlds at time t' . Let the variable r range over these equivalence classes. The r form a partition of W , so the probability of a set X can be written as the integral over this partition:

$$\mu_t^w(X) = \int_{r \in W} \mu_t^r(X|r) \mu_t^r(dr)$$

Since the history up to time t' determines the probability at time t' , this can be written as

$$\mu_t^w(X) = \int_{r \in W} \mu_{t'}^r(X) \mu_t^r(dr),$$

where $\mu_{t'}^r$ denotes the probability at time t' in equivalence class r . Since the probability at a given time is assumed to be constant over all worlds in an R-equivalence class, the probability at a given time is the expected value of the probability at any future time:

$$\mu_t^w(X) = \int_W \mu_{t'}^r(X) \mu_t^r(dw).$$

Next we show that Miller's principle is valid in the probability models. By the expected value property,

$$\begin{aligned} \mu_t^w(X \cap \{w' : \mu_{t'}^r(X) = \alpha\}) &= \\ \int_W \mu_{t'}^r(X \cap \{w' : \mu_{t'}^r(X) = \alpha\}) \mu_t^r(dw'). \end{aligned}$$

Now, by semantic constraints (C6) and (C7) it follows that

$$\begin{aligned} \forall w \in \{w' : \mu_{t'}^r(X) = \alpha\}, \mu_t^r(\{w' : \mu_{t'}^r(X) = \alpha\}) &= 1 \\ \forall w \notin \{w' : \mu_{t'}^r(X) = \alpha\}, \mu_t^r(\{w' : \mu_{t'}^r(X) = \alpha\}) &= 0. \end{aligned}$$

So we can restrict the integral to the set $\{w' : \mu_{t'}^r(X) = \alpha\}$:

$$= \int_{\{w' : \mu_{t'}^r(X) = \alpha\}} \mu_{t'}^r(X \cap \{w' : \mu_{t'}^r(X) = \alpha\}) \mu_t^r(dw').$$

And by the above property again

$$\mu_{t'}^r(X \cap \{w' : \mu_{t'}^r(X) = \alpha\}) = \alpha, \text{ so}$$

$$\begin{aligned} &= \alpha \cdot \int_{\{w' : \mu_{t'}^r(X) = \alpha\}} \mu_t^r(dw') \\ &= \alpha \cdot \mu_t^r(\{w' : \mu_{t'}^r(X) = \alpha\}). \end{aligned}$$

By the semantic definitions it follows that

$$P_t(\phi \wedge P_{t'}(\phi) = \alpha) = \alpha \cdot P_t(P_{t'}(\phi) = \alpha).$$

And by a slight generalization of the proof it follows that

$$\forall (t \preceq t') P_t(\phi \wedge P_{t'}(\phi) \geq \alpha) \geq \alpha \cdot P_t(P_{t'}(\phi) \geq \alpha).$$

Acknowledgements

This work was supported in part by the author's Cognitive Science/Artificial Intelligence fellowship from the University of Illinois and in part by the author's Shell doctoral dissertation fellowship. The author would like to thank Alan Frisch, Patrick Maher, David Page, Richard Scherl, Tomas Uribe, and Yong Ma for helpful comments on earlier drafts of this paper.

References

- [Allen, 1984] J.F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.
- [Bacchus, 1988] F. Bacchus. *Representing and Reasoning With Probabilistic Knowledge*. PhD thesis, University of Alberta, 1988.
- [Fagin et al., 1988] R. Fagin, J.Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. Technical Report RJ 6190 (60900), IBM Almaden Research Center, April 1988. (also to appear in *Information and Computation*).
- [Gaifman, 1988] H. Gaifman. A theory of higher order probabilities. In B. Skyrms and W.L. Harper, editors, *Causation, Chance, and Credence*, pages 191–219. Kluwer Academic Publishers, 1988.
- [Haas, 1985] A.R. Haas. Possible events, actual events, and robots. *Computational Intelligence*, 1:59–70, 1985.
- [Haddawy and Frisch, 1990] P. Haddawy and A.M. Frisch. Modal logics of higher-order probability. In R. Shachter, T.S. Levitt, J. Lemmer, and L.N. Kanal, editors, *Uncertainty in Artificial Intelligence 4*, pages 133–148. Elsevier Science Publishers, Amsterdam, 1990.
- [Haddawy, 1990] P. Haddawy. Time, chance, and action. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 147–154, Boston, July 1990.
- [Haddawy, 1991] P. Haddawy. *Representing and Reasoning About Plans Under Uncertainty*. PhD thesis, University of Illinois, 1991. (to appear).
- [Halpern and Tuttle, 1989] J.Y. Halpern and M.R. Tuttle. Knowledge, probability, and adversaries. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, pages 103–118, August 1989.
- [Halpern, 1989] J. Y. Halpern. The relationship between knowledge, belief, and certainty. In *Proceedings of the Fifth Workshop on Uncertainty in AI*, pages 142–151, 1989. An expanded version appears as IBM Research Report RJ 6765, 1989.
- [Halpern, 1991] J.Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1991.
- [Hart and Shair, 1984] S. Hart and M. Shair. Probabilistic temporal logics for finite and bounded models. In *Proceedings of the Sixteenth ACM Symposium on the Theory of Computing*, pages 1–13, 1984.
- [Lehmann and Shelah, 1982] D. Lehmann and S. Shelah. Reasoning with time and chance. *Information and Control*, 53:165–198, 1982.
- [Lewis, 1980] D. Lewis. A subjectivist's guide to objective chance. In W. Harper, R. Stalnaker, and G. Pearce, editors, *Ifs*, pages 267–298. D. Reidel, Dordrecht, 1980.
- [Lewis, 1981] D. Lewis. Causal decision theory. *Australasian Journal of Philosophy*, 56(1):5–30, March 1981.
- [Maher, 1987] P. Maher. Causality in the logic of decision. *Theory and Decision*, 22:155–172, 1987.
- [McDermott, 1982] D.V. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.
- [Mellor, 1987] D.H. Mellor. Fixed past, unfixed future. In B.M. Taylor, editor, *Michael Dummett*, pages 166–186. Martinus Nijhoff Publishers. Dordrecht, Netherlands, 1987.
- [Moore, 1985] R.C. Moore. A formal theory of knowledge and action. In J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*. Ablex, Norwood, NJ, 1985.
- [Pelavin and Allen, 1986] R.N. Pelavin and J.F. Allen. A formal logic of plans in temporally rich domains. *Proceedings of the IEEE*, 74(10):1364–1382, October 1986.
- [Pelavin, 1988] R.N. Pelavin. *A Formal Logic for Planning with Concurrent Actions and External Events*. PhD thesis, Univ. of Rochester, Dept. of Computer Science, 1988.
- [Shoham, 1987] Y. Shoham. *Reasoning About Change*. MIT Press, Cambridge, MA, 1987.
- [Shoham, 1989] Y. Shoham. Time for action: On the relationship between time, knowledge, and action. In *Proceeding of the Eleventh International Joint Conference on Artificial Intelligence*, pages 954–959 & 1173, Detroit, August 1989.
- [Skyrms, 1980a] B. Skyrms. *Causal Necessity*. Yale Univ. Press, New Haven, 1980.
- [Skyrms, 1980b] B. Skyrms. Higher order degrees of belief. In D.H. Mellor, editor, *Prospects for Pragmatism*, chapter 6, pages 109–137. Cambridge Univ. Press, Cambridge, 1980.
- [Suppes, 1970] P. Suppes. *A Probabilistic Theory of Causality*. Acta Philosophica Fennica, Fasc. XXIV. North-Holland, Amsterdam, 1970.
- [van Fraassen, 1980] B.C. van Fraassen. A temporal framework for conditionals and chance. In W. Harper, R. Stalnaker, and G. Pearce, editors, *Ifs*, pages 323–340. D. Reidel, Dordrecht, 1980.
- [Venn, 1866] J. Venn. *The Logic of Chance*. MacMillan, London, 1866. (new paperback edition, Chelsea, 1962).
- [von Mises, 1957] R. von Mises. *Probability, Statistics and Truth*. Allen and Unwin, London, 1957.

Model Checking vs. Theorem Proving: A Manifesto

Joseph Y. Halpern
 IBM Almaden Research Center
 650 Harry Road
 San Jose, CA 95120
 email: halpern@ibm.com

Moshe Y. Vardi
 IBM Almaden Research Center
 650 Harry Road
 San Jose, CA 95120
 email: vardi@ibm.com

Abstract

We argue that rather than representing an agent's knowledge as a collection of formulas, and then doing theorem proving to see if a given formula follows from an agent's knowledge base, it may be more useful to represent this knowledge by a semantic model, and then do model checking to see if the given formula is true in that model. We discuss how to construct a model that represents an agent's knowledge in a number of different contexts, and then consider how to approach the model-checking problem.

1 INTRODUCTION

The standard approach in AI knowledge representation, going back to [McCarthy, 1968], is to represent an agent's knowledge as a collection of formulas, which we can view as a *knowledge base*. An agent is then said to know a fact if it is *provable* from the formulas in his knowledge base. This is called in [Rosenschein, 1985] the "interpreted-symbolic-structures" approach. There are two problems in applying this approach. The first comes in the difficulty of representing agents' knowledge in terms of formulas in some appropriate language. The second lies in the difficulty of theorem proving. These problems are closely related; the need to use logic to represent agents' knowledge necessitates the use of very expressive logics, but the more expressive a logic, the harder it is to prove theorems in that logic. In this paper, we argue for a model-theoretic rather than a proof-theoretic approach to the problem. Essentially, the idea is to represent the agent's knowledge by a data structure representing some semantic model (in the spirit of the "situated-automata" approach [Rosenschein, 1985; Rosenschein and Kaelbling, 1986]), and replace theorem proving by *model checking*, that is, checking whether a given formula is true in the model.

As an example of this approach consider the con-

text of relational database systems. Let B be a relational database and let φ be a first-order query. The theorem-proving approach would view B as representing some formula φ_B and would evaluate the query by trying to prove or disprove $\varphi_B \Rightarrow \varphi$. Unfortunately, theorem proving for first-order logic is undecidable. The model-checking approach, on the other hand, would check whether φ holds in the database B . This can be evaluated in time polynomial in the size of the data (cf. [Vardi, 1982]).

As another example, consider an agent Alice who knows that, given her current epistemic state (i.e., the information she has obtained thus far), the world could be in any one of three possible states. In the possible-worlds approach this situation is modelled by a Kripke structure with three possible worlds. As usual, we say that Alice knows a fact p if p is true in all three of the worlds that Alice considers possible given her epistemic state.

Since, in particular, all tautologies will be true at all three worlds Alice considers possible, it follows that Alice knows all propositional tautologies. This may seem strange. The set of propositional tautologies is well-known to be co-NP-complete. How can Alice, who after all does not seem to possess any extraordinary reasoning power, know all tautologies when she can't prove them? In the model-checking approach, there is nothing unusual about this fact. Alice knows a propositional formula φ if φ is true in the three states that Alice considers possible. This can be checked in time linear in the length of φ . Notice that even though Alice knows all tautologies (among other facts she knows given her epistemic state), she does not know which of the facts she knows are tautologies (and probably does not care!). Thus, the well-known *logical omniscience* problem [Hintikka, 1975] does not present the same difficulties in the model-checking approach as it does in the theorem-proving approach (although, as we shall see, other related difficulties do arise).

The paradigm of model-checking arose explicitly in the context of finite-state program verification (see [Clarke

and Grümberg, 1987] for an overview). Suppose we have a finite-state program P (think of P as, for example, a communications protocol), and we want to know whether it satisfies some specification ψ , which we assume can be expressed in temporal logic.¹ It is not hard to completely characterize the program P by a temporal logic formula φ_P (essentially, φ_P describes all the possible transitions of P in each possible global state; this is possible since P is a finite-state protocol). One way of checking whether P satisfies the specification ψ is to check if $\varphi_P \Rightarrow \psi$ is valid [Manna and Pnueli, 1981]. Unfortunately, the validity problem for temporal logic is extremely difficult (technically, it is exponential-time complete, which most likely makes it much harder than the validity problem for propositional logic [Emerson and Halpern, 1985]).

Later, researchers noticed that another approach would work equally well [Clarke *et al.*, 1986; Queille and Sifakis, 1981]. Rather than having P be represented by a formula, P can be represented by a Kripke structure M_P : the states in M_P represent the possible global states of P , and the edges represent the possible transitions of P . Note that the size (i.e., the number of states) of M_P is essentially the same as the length of φ_P (viewed as a string of symbols). Checking whether P satisfies the specification ψ now amounts to checking if ψ is true at the state in M_P that corresponds to the initial state of P . This can be done in time linear in the size of M_P and φ .

Of course, it could be argued that all that this argument shows is that for a special subclass of formulas (namely, those of the form $\varphi_P \Rightarrow \psi$), the validity problem is significantly simpler than it is in general. Indeed, perhaps this observation should encourage us to find other subclasses for which the validity problem is solvable in polynomial time (cf. [Emerson *et al.*, 1989]). However, we would claim that this argument misses the point. The reason that formulas of the form $\varphi_P \Rightarrow \psi$ are easy to deal with is because φ_P characterizes a particular state in a particular structure, in that any state where φ_P is satisfied is isomorphic to the initial state of the structure defined by P . Thus, for these formulas, the validity problem reduces to a model-checking problem, and so is tractable.

The model-theoretic approach to finite-state program verification is quite practical, and several systems based on model checking have been implemented [Clarke and Grümberg, 1987; Burch *et al.*, 1990]. Moreover, it has been extended to deal with more complicated protocols and environments (including probabilistic protocols and assumptions of fairness [Vardi and Wolper, 1986]). In some cases, the assumptions

¹Our discussion here presumes the use of branching temporal logic. In case of linear temporal logic, the situation is somewhat more complicated, see [Emerson and Halpern, 1985; Sistla and Clarke, 1985; Lichtenstein and Pnueli, 1985].

that are being dealt with (such as fairness) are not even expressible in the language being used for the specification formula. This is not a hindrance to the model-theoretic approach. We can often check whether the model satisfies assumptions that are not expressible in the language.

In this paper, we describe the model-checking approach in detail and relate it to a number of current trends in AI research. We also discuss potential problems with the approach, show how some might be dealt with by using current techniques, and suggest some lines of research for dealing with others. We also consider a logic appropriate for reasoning using the model-checking approach, inspired by the logic of resource-bounded knowledge presented in [Moses, 1988].

2 USING THE MODEL-THEORETIC APPROACH

Clearly, if we are to apply the model-theoretic approach, two questions need to be dealt with:

1. What is the appropriate semantic model?
2. How can we check if a formula is true in the model?

It seems best to tackle these questions separately.

2.1 WHAT IS THE APPROPRIATE SEMANTIC MODEL?

Many AI applications deal with agents interacting among themselves and/or with an external environment.² In such a setting, we believe that the possible-worlds paradigm yields the appropriate semantic model. Moreover, in analyzing such systems, we claim that it is often helpful to be able to reason not only at the assertional level, but also about the knowledge of agents in the system (see [Halpern, 1986] for an overview of this approach). In order to do this, we start with an *assertion language* (typically a first-order or propositional language) that describes basic properties of the system and then extend it with modalities K_i for each agent i . Intuitively, $K_i\varphi$ holds if agent i knows φ .

The traditional way to capture this in a formal framework is to use *possible worlds* or *Kripke structures*. Recall that a Kripke structure M for n agents is a tuple $(S, \pi, K_1, \dots, K_n)$, where S is a set of possible worlds

²The notion of "agent" should be taken rather loosely here. An agent can be a robot observing an environment, a knowledge base (or knowledge bases) being told information, or a processor in a parallel machine. Everything we say applies in all of these contexts.

or *states*, π associates with each world in S a truth assignment (propositional or first-order as the case may be) and K_i is a *binary relation* on S . Intuitively, the truth assignment $\pi(w)$ tells us what is true in a world w . The relation K_i is intended to capture the possibility relation according to agent i : $(u, v) \in K_i$ if agent i considers world v possible when in world u . We then say agent i knows φ at world u in structure M , and write $(M, u) \models K_i \varphi$, if φ is true in all the worlds v that agent i considers possible in world u . This captures the intuition that an agent knows a fact if it is true at all the worlds he considers possible.

We would like to model a system of interacting agents as a Kripke structure. To do so, we use the formal model of [Fagin et al., 1986; Halpern and Fagin, 1989] (which in turn is based on earlier models which appeared in [Halpern and Moses, 1990; Parikh and Ramanujam, 1985; Chandy and Misra, 1986; Rosenschein and Kaelbling, 1986]). We briefly describe the formal model here, since we shall make use of it later, referring the reader to [Halpern and Fagin, 1989] for more details.

We assume that at each point in time, each agent is in some *local state*. Informally, this local state encodes the information it has observed thus far. In addition, there is also an *environment state*, that keeps track of everything relevant to the system not recorded in the agents' states. A *global state* is a sequence (s_e, s_1, \dots, s_n) consisting of the environment state s_e and the local state s_i of each agent i . A *run* of the system is a function from time (which, for ease of exposition, we assume ranges over the natural numbers, although we could easily take it to range over the reals) to global states. Thus, if r is a run, then $r(0), r(1), \dots$ is a sequence of global states that, roughly speaking, is a complete description of what happens over time in one possible execution of the system. We take a system to consist of a set of runs. Intuitively, these runs describe all the possible sequences of events that could occur in a system.

Given a system \mathcal{R} , we refer to a pair (r, m) consisting of a run $r \in \mathcal{R}$ and a time m as a *point*. The points can be viewed as worlds in a Kripke structure. We say two points (r, m) and (r', m') such that $r(m) = (s_e, s_1, \dots, s_m)$ and $r'(m') = (s'_e, s'_1, \dots, s'_{m'})$ are *indistinguishable* to agent i , and write $(r, m) \sim_i (r', m')$, if $s_i = s'_i$, i.e., if agent i has the same local state at both points. We take \sim_i to play the role of the possibility relation K_i ; note that K_i is an equivalence relation under this interpretation.

An *interpreted system* is a pair (\mathcal{R}, π) consisting of a system \mathcal{R} together with a mapping π that associates a truth assignment with each point. The semantics of knowledge formulas in interpreted systems is identical to that in Kripke structures. In particular, given a point (r, m) in an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$, we have $(\mathcal{I}, r, m) \models K_i \varphi$ if $(\mathcal{I}, r', m') \models \varphi$ for all points

(r', m') such that $(r', m') \sim_i (r, m)$. Notice that under this interpretation, an agent knows φ if φ is true at all the situations the system could be in, given the agent's current information (as encoded by its local state).

For a given distributed protocol, it is often relatively straightforward to construct the system corresponding to the protocol. The local state of each process can typically be characterized by a number of internal variables (that, for example, describe the messages thus far received and the values of certain local variables), and there is a transition function that describes how the system changes from one global state to another. (See [Halpern and Zuck, 1987] for a detailed example of the modelling process.) For knowledge-based applications, the modeling problem may be harder because of the difficulty in describing the state space. Basically, it is not clear how to describe the states when they contain information about agents' knowledge. In this context, the approach suggested in [Fagin et al., 1984] may prove useful cf. [Hamilton and Delgrande, 1989; Lejoly and Minsoul, 1990].

Suppose we have constructed a model of the system. Provided that we can completely characterize an agent's local state s by a formula φ_s , then we can reduce the problem of checking whether an agent in local state s knows φ to checking the validity of $\varphi_s \Rightarrow \varphi$, since agent i knows φ in local state s if φ is true at all possible worlds where its local state is s . However, being able to characterize an agent's local state by means of a formula will almost certainly require the use of a very expressive logic, for which theorem proving is quite intractable. The model-checking approach, on the other hand, does not require that local states be encodable as formulas. Thus, it does not require resorting to a logic that is more expressive than necessary to express the assertion φ .

An important issue to consider when comparing the two approaches is that of representation. The theorem-proving approach requires us to represent the agent's knowledge by a collection of formulas in some language. The model-checking approach instead represents the agent's knowledge as a local state in some structure. We have, however, complete freedom to decide on the representation of the local state and the structure. Indeed, there are applications that arise frequently in AI where the logical choice for the representation is in terms of formulas. For example, the problem-solving system STRIPS represents a state by a set of first-order formulas; operations on the state are represented by adding certain formulas and deleting others [Fikes and Nilsson, 1972]. As another example, Levesque [Levesque, 1984a] studies knowledge bases (KBs) where we have *TELL* and *ASK* operations. After the KB is told a sequence of facts, we should be able to represent the situation using the set

κ of facts that it has been told.³ ⁴ What Kripke structure does this represent?

The first thing we have to decide is what the global states are going to be. There is only one agent in the system, namely, the KB itself, so a global state will be a pair consisting of the environment state and the KB's local state. As we said above, we can identify the KB's local state with the formulas that it has been told. What about the environment? Since we assume that there is a real world external to the KB, we can take the environment to be a complete description of the relevant features of the world. We model this by taking the environment state to be a *relational structure*.⁵ Thus, a global state is a pair (A, κ) , where A is a relational structure (intuitively, describing the world) and κ is a formula (intuitively, the conjunction of the facts the KB has been told).

Next we have to decide what the allowable global states are. This turns out to be quite sensitive to the expressiveness of the KB. Suppose we assume that the KB is *assertion-based*, i.e., it is told only facts in the assertion language and no facts about its knowledge. Then we can restrict attention to global states (A, κ) where the formula κ is true in A (this captures the assumption that the KB is only told true facts about the world). If we make no further restrictions, then, according to our definition of knowledge, the knowledge base KB *knows* a fact φ in a state κ if φ holds in all the global states of the form (A, κ) . But that means that the KB knows φ in a state κ if φ holds

³To model the evolution of the knowledge base, one has to use some temporal structure where each point is represented by a set of formulas. See [Morris and Nado, 1986] and references therein for a discussion of such a model.

⁴Although it may indeed be reasonable in many cases to represent a situation by the set of facts an agent has been told, note that by doing so we are assuming (among other things) that the KB is being told facts about a static situation, so that when the information arrived is irrelevant. For example, if we consider propositions whose truth may change over time, it may be that at some point the KB is told φ and then later it is told $\neg\varphi$. We do not necessarily want to view the KB as being in an inconsistent state at this point. We can get around this difficulty by augmenting our representation to include time, so that " φ at time 3" would not be inconsistent with " $\neg\varphi$ at time 4". We can also imagine a more sophisticated KB, that discards an earlier fact if it is inconsistent with later information. However, such an approach quickly leads to all the difficulties one encounters in general with updating beliefs (see, e.g., [Fagin et al., 1983; Gardenfors, 1988]). For example, what do we do if we have three facts which are pairwise consistent, but whose conjunction is inconsistent? Which of the three facts do we discard?

⁵A relational structure consists of a domain of individual elements, and an assignment of functions and relations to the function and relation symbols of the assertion language.

in all the models of κ or, equivalently, if φ is a logical consequence of κ .

This example already shows how the model-checking approach can be viewed as a generalization of the theorem-proving approach. But the model-checking approach gives us added flexibility. For example, consider the *closed-world assumption* [Reiter, 1984], where any fact not explicitly stored in the database is taken to be false. The need for such an assumption about negative information stems from the fact that in any complex application, the number of negative facts vastly outnumbers the number of positive ones, so that it is totally infeasible to explicitly represent the negative information in the database. We can capture this assumption model-theoretically by restricting attention to pairs (A, κ) where not only is κ true in A , but every atomic formula $P(d_1, \dots, d_n)$ not implied by κ is false in A . While the closed-world assumption can be formalized in first-order logic, which means that we can use the theorem-proving approach to query closed-world knowledge bases [Reiter, 1984], analyzing the problem from the model-checking perspective leads to a complete characterization of the complexity of query answering, and furthermore, it enables one to deal with higher-order queries, that are not usually amenable to the theorem-proving approach [Vardi, 1986].

The closed-world assumption is only one of many we might consider making to restrict attention to only certain (A, κ) . As Shoham [Shoham, 1987] observed, most forms of nonmonotonic reasoning can be viewed as attempts to restrict attention to some collection of preferred pairs. The focus in many of the works on nonmonotonic reasoning is on using some logical formalism to describe the set of preferred models. This is necessary if one wishes to apply the theorem-proving approach. Our contention is that theorem proving is just one way to evaluate queries and not necessarily the optimal way. From our point of view, the right question is not whether one can prove if an assertion φ follows from a knowledge base κ , using some nonmonotonic logic, but rather if φ holds in the Kripke structure represented by κ (where the nonmonotonicity is captured by restricting attention to a preferred set of possible worlds).

Even without the complication arising from the notion of preference, the situation gets more complicated if we assume that the facts that the KB is told include information about its own knowledge, i.e., the specification is *knowledge-based*. As pointed out in [Levesque, 1984a], this can be quite important in practice. For example, suppose a KB is told the following facts about a small group of people: "John is a teacher", "Mary is a teacher", and "you know about all the teachers". Then, we expect the KB to know that Bill is not a teacher (assuming it knows that Bill is distinct from Mary and John). Intuitively, this is because the three assertions together restrict the set of possible worlds

to ones where only John and Mary are teachers.

How can we model this? That is, what pairs (A, κ) should we allow now (where κ is the conjunction of the three formulas mentioned above)? Clearly, taking pairs (A, κ) such that κ is true in A will not work, since a relational structure A does not determine the truth of formulas involving knowledge. The answer is that when we have a knowledge-based specification, we cannot select the allowable states one at a time, even if we allow some notion of preference. Rather, it is the collection of all possible states that has to be consistent with the KB. Thus, we cannot just focus on the global states; we have to consider the whole resulting Kripke structure. Intuitively, we are trying to describe a Kripke structure consisting of a collection of pairs (A, κ) such that κ is true in the resulting Kripke structure (notice that the truth of κ depends on the whole Kripke structure, not just on A).

Just as there are potentially many states consistent with a given assertion-based specification, there are potentially many Kripke structures consistent with a given knowledge-based specification. We can use some notion of preference to select a unique Kripke structure. A popular notion of preference is one that tries to circumscribe the KB's knowledge. The idea is to view the knowledge specified in the KB as *all that is known*. Thus, this preference can be viewed as a closed-world assumption at the knowledge level. This notion has received a lot of attention in the last decade [Fagin et al., 1984; Halpern and Moses, 1984; Konolige, 1984; Levesque, 1981; Levesque, 1984a; Levesque, 1987; Parikh, 1984; Vardi, 1985].

So far we have considered only situations with a single agent. In such situations, Kripke structures degenerate to essentially sets of worlds. Many AI applications, however, deal with multiple interacting agents. It is when we try to give formal semantics to sentences such as "Dean doesn't know whether Nixon knows that Dean knows that Nixon knows about the Watergate break-in" that the full power of Kripke structures come into play. In such situations circumscribing the agents' knowledge is highly nontrivial (cf. [Fagin et al., 1984; Parikh, 1984]).

To summarize, in relatively simple settings it may not be too difficult to describe, implicitly or explicitly, the set of runs that characterize the system, and thus we can get a Kripke structure, which is the appropriate semantic model. Preference criteria and defaults may make this task more difficult, but we expect that even in this case, the task will become manageable as our understanding of these notions deepens.

2.2 Model checking

Suppose we have somehow constructed what we consider to be the appropriate semantic model. We now want to check if a given formula φ is true in a partic-

ular state of that model.

Checking whether an arbitrary formula in the extended language holds in a given world of a given Kripke structure has two components: we have to be able to check whether *assertions* (i.e., knowledge-free formulas) hold in a given state (we call this *assertion checking*) and check whether *knowledge* formulas (i.e., formulas of the form $K_i\varphi$) hold in a given state. If we can do both of these tasks, then we can handle arbitrary formulas by induction on the structure of the formula. Since checking whether a knowledge formula holds involves quantification over the possible worlds, the complexity of model checking depends on (and is typically polynomial in the product of) three quantities: the complexity of assertion checking, the size of the given Kripke structure, and the size of the formula. Thus, our ability to do model checking efficiently depends crucially on our ability to do assertion checking efficiently and our ability to deal with "large" structures.

For propositional languages, assertion checking is quite easy: it can be done in linear time. This is not the case for first-order formulas. If we consider structures with infinite domains, it is not always clear how to represent them, let alone do assertion checking. Even if we restrict to structures with finite domains, assertion checking may be very difficult due to the size of the domain. In general, in order to check the truth of a formula such as $\forall x\varphi(x)$, we may have to check the truth of $\varphi(d)$ for each domain element d .

Even if assertion checking is easy, we still have to cope with the multitude of possible worlds. For example, in the context of incomplete knowledge bases, the number of possible worlds can be exponential in the size of the knowledge base, which makes query evaluation intractable [Vardi, 1986]. We note, however, that the fact that the number of possible worlds is quite large does not automatically mean that it is hard to check all knowledge formulas. The results of [Dwork and Moses, 1990; Moses and Tuttle, 1988] demonstrate that some knowledge formulas of interest can be evaluated efficiently in certain contexts despite an exponential number of possible worlds.

What can we do to deal with situations where the number of possible worlds or the number of domain elements is too large to handle? There are a number of approaches one could pursue. For one thing, we might hope to be able to find restricted (but still interesting) subclasses of formulas for which we can do model checking efficiently in certain restricted (but still interesting) subclasses of structures, just as we now have techniques for doing inference efficiently with certain subclasses of formulas (such as Horn clauses). Indeed, IS-A formulas (which essentially correspond to Horn clauses, although see [Brachman, 1985] for some caveats) can be checked very efficiently in IS-A hierarchies.

A second approach is to consider heuristics and defaults. Since the difficulty in model-checking arises when we have large structures and/or large domains, we need good heuristics for handling such situations. We in fact use such heuristics all the time in our daily life. For example, if there are too many possibilities (i.e., too many possible worlds), we focus attention on only a few (the ones we deem to be the “most relevant” according to some metric or “most likely” according to some probability distribution), ignoring the rest. Consider a situation where Alice is asked if (she knows that) φ is the case. In order to answer this question, Alice would have to check all the worlds she considers possible to see if φ holds in all of them. There may be 1,000,000 worlds that Alice considers possible, which means that she has a lot of checking to do. However, if Alice has a probability space on these worlds, then there may be a subset of worlds of size 1,000 that has probability .99. Alice could check these 1,000 worlds; if φ holds in all of them, then this might be taken as sufficient “evidence” for Alice to say that φ indeed holds. Even without explicit probability, Alice might still have a notion of which of the 1,000,000 worlds are most likely or most relevant. Similarly, if we have a large domain and want to check the truth of a formula such as $\forall x\varphi(x)$, we can use heuristics and defaults to cut down the search space (perhaps checking $\varphi(d)$ for only a few potentially “abnormal” domain elements).

Yet another approach would be to have a notion of what features are “relevant” to the problem, and identify all worlds that agree on irrelevant features, thus cutting down on the search space. Being able to do this depends both on being able to identify relevant notions and on being able to check relevant features rapidly. Going back to our previous example, suppose φ is a propositional formula such as $(p_1 \vee p_2) \wedge (\neg p_2 \vee p_3)$, where p_1 , p_2 , and p_3 are primitive propositions. It is easy to see that φ is true unless p_1 and at least one of p_2 or p_3 are false. The truth values of all other propositions are clearly irrelevant to the truth of φ . Depending on how Alice’s possible worlds are represented, it may be that Alice can check that Alice does not consider a world possible where p_1 and at least one of p_2 and p_3 are false, without checking all 1,000,000 possible worlds. We remark that formalisms for expressing and reasoning with irrelevance have been developed, both probabilistic (see, for example, [Pearl and Verma, 1987]) and non-probabilistic [Subramanian and Genesereth, 1987]), but this is an area where much further work remains to be done.

As all these approaches show, much depends on precisely how the worlds are generated and represented. Finding appropriate representations is, of course, a major open problem.

A great deal of effort in theorem proving has been expended on finding heuristics that work well for the formulas that arise in practice. Analogously, in model

checking, it would be useful to find heuristics that work well for structures that arise in practice. Although the work on model checking in this regard is still in its infancy, the early results appear quite promising [Holzmann, 1988; Burch *et al.*, 1990]. For example, when verifying circuits, we are not dealing with arbitrary Kripke structures. The regularity in the structure suggests heuristics that seem to work well in structures with up to 10^{20} states [Burch *et al.*, 1990]!

Of course, techniques need to be developed in order to allow us to use these heuristics and defaults in a principled way. Much of the recent work in “vivifying” can be viewed as being in this spirit [Etherington *et al.*, 1989; Levesque, 1986]. The idea is to cut down on the number of possibilities by filling-in some (hopefully) inessential details. Work on a possible-worlds framework for probability [Bacchus, 1990; Fagin and Halpern, 1989; Halpern, 1990; Nilsson, 1986] may provide insights into using probabilistic heuristics in a principled way to cut down the search space.

Notice that by using defaults in this way, we are led naturally to nonmonotonicity. However, rather than the logic being nonmonotonic, the model checking is nonmonotonic. We might withdraw some of our conclusions about a formula being true in a given structure if we get further information that leads us to believe that the default assumptions we used to simplify the model-checking problem are not correct.

2.3 Modelling resource-bounded agents

Agents must act on the basis of their knowledge. In fact, a good way of describing the planning process is often in terms of what agents need to know in order to carry out their actions [Moore, 1985]. In the context of distributed systems, there have been a number of papers giving *knowledge-based* communication protocols, where actions explicitly depend on the outcomes of tests for knowledge (see, e.g., [Halpern and Zuck, 1987]). Our definition of knowledge is information-theoretic; roughly speaking, an agent knows a fact if it holds in all states where the agent has the same information as in the current state. This definition does not take computation into account.

When we talk about “computing knowledge” here, what we are really talking about is the agent performing model checking. The agent is certainly not interested in computing whether φ is valid; very few actions depend on valid preconditions. Nevertheless, model-checking can also be intractable. It can still be the case, given our definition, that an agent can “know” a formula without being able to compute that it knows the formula. This can be viewed as the reincarnation of the logical omniscience problem in the model-checking framework.

We would like to have a formal way of capturing the

knowledge of a resource-bounded agent in the model-checking framework. In [Moses, 1988], Moses presents a logic of resource-bounded knowledge, where he tries to make sense out of notions such as “an agent can compute φ in polynomial time”. His approach is very much in the spirit of the model-theoretic approach advocated here, since what it means for him to be able to compute φ in polynomial time is *not* that φ can be proved (in some appropriate axiom system) in polynomial time, but rather that the truth of φ at a particular state in a structure can be computed in polynomial time. We informally discuss some details of Moses’ logic here and relate it to our framework; the reader is encouraged to consult [Moses, 1988] for more details and motivation.

Moses uses the distributed systems model discussed in the previous section, where a system is identified with a set of runs. As we mentioned above, the definition of what it means for $K_i\varphi$ to hold at a point in a run does not take into account complexity-theoretic considerations. We want to define a notion $K_i^P\varphi$ which intuitively amounts to “agent i knows φ and, moreover, can compute this knowledge in polynomial time.”⁶ Notice that whether $(\mathcal{I}, r, m) \models K_i\varphi$ depends only on $r_i(m)$, agent i ’s local state at the point (r, m) . If we say that agent i knows how to compute φ in polynomial time, we take this to mean that *no matter what state i is in, i can compute φ* . In order to capture this intuition, we require that there exist an algorithm A that gets i ’s local state as input, and computes in polynomial time whether φ is implied by the local state. Notice that there is a slight subtlety here: we have said that A must run in polynomial time, but we have not said what the time is polynomial in. We sidestep this issue by simply assuming that with each local state, there is some parameter; the computation must be polynomial in that parameter. For example, the parameter can be the number of agents in the system, a particular shared input (this is appropriate in cryptographic applications; see [Halpern et al., 1988]), or the time (since this corresponds roughly to the number of pieces of information that the agent has received thus far, so the computation would be polynomial in the amount of information held by the agent). To summarise, given an interpreted system \mathcal{I} and a point (r, m) in the system, we say that $(\mathcal{I}, r, m) \models K_i^P\varphi$ if

⁶We remark that polynomial time knowledge can be viewed as a special case of explicit knowledge in the logic of general awareness of [Fagin and Halpern, 1988]. In the logic of general awareness, there are operators K_i , B_i , and A_i , where $K_i\varphi$, $B_i\varphi$ and $A_i\varphi$ represent, respectively, that agent i implicitly knows φ , explicitly knows φ , and is aware of φ . An agent explicitly knows φ exactly if he implicitly knows φ and is aware of φ ; thus, the equivalence $B_i\varphi \equiv (K_i\varphi \wedge A_i\varphi)$ holds. In the logic of resource-bounded reasoning, polynomial-time knowledge (K_i^P) plays the role of explicit knowledge, the usual notion of knowledge (K_i) plays the role of implicit knowledge, and “being able to compute in polynomial time” plays the role of awareness.

1. $(\mathcal{I}, r, m) \models K_i\varphi$, and
2. there exists an algorithm A that takes as input s , a local state of agent i , and returns “Yes” or “No”, depending on whether $K_i\varphi$ holds in all (resp. none) of the points where i has local state s ; moreover, A runs in time polynomial in the parameter associated with s .

The logic of resource-bounded reasoning provides an elegant framework for analyzing resource-bounded notions of knowledge. It is shown to be useful for analyzing distributed protocols in [Moses, 1988], and it is extended in a number of ways in [Halpern et al., 1988] to deal with cryptographic protocols. Some of these extensions suggest further modifications which make the logic more applicable to AI. We discuss these here.

First note that if $K_i^P\varphi$ holds, then there is *some* polynomial time algorithm that will allow the agent to compute whether he knows φ at any state in the structure. However, the agent may not know what that algorithm is. More realistically, rather than having access to all polynomial time algorithms, an agent may have several algorithms that she can try to use to figure out whether she knows φ . It is easy to modify the logic to handle this situation. Let \mathcal{B} be some set of algorithms. We then define $(\mathcal{I}, r, m) \models K_i^{\mathcal{B}}\varphi$ just as we did $(\mathcal{I}, r, m) \models K_i^P\varphi$, except that in the second clause, we require that $A \in \mathcal{B}$, rather than that A is polynomial time. (Of course, if we take \mathcal{B} to consist of all polynomial-time algorithms, then we recover the definition of $K_i^P\varphi$.) Thus, if Alice tries exactly one of two algorithms to figure out if φ is true, then \mathcal{B} would consist of those two algorithms, and $K_{Alice}^{\mathcal{B}}\varphi$ would hold at a point exactly if Alice knows φ and she can compute this fact using one of her two algorithms. We can imagine that different agents have access to different algorithms; an agent with more expertise would have access to more and better algorithms. In this way, we can model a situation where one agent knows how to do something while another does not.

It may seem to be too strong a requirement for Alice to have to *know* φ . In practice, it may be enough for Alice to know φ with high probability, or to let her algorithms make occasional mistakes, as long as they are not too frequent. It is not too hard to modify our definitions to get such probabilistic notions of computable knowledge; these issues are discussed in more detail in [Halpern et al., 1988]. These probabilistic notions where error is allowed may give us a principled way to deal with the problem of having too many states in the model to do efficient model checking. This issue deserves further exploration.

3 CONCLUSIONS

We have argued here for a model-theoretic rather than a proof-theoretic approach to reasoning about knowl-

edge. We do not mean to suggest that the model-theoretic approach is a panacea. There are difficulties to be overcome here, involving how to find the model and how to do model-checking on large structures, just as there are difficulties in the theorem-proving approach. Our hope is that the model-theoretic perspective will suggest new heuristics and new approaches. We feel that many of the approaches that are currently being tried, including the preferred models approach to nonmonotonic reasoning and the idea of "vivifying" knowledge bases, are also best understood in the context of the model-checking framework. Thus, this perspective may also allow us to unify a number of current lines of research.

We do not expect the model-checking approach to supplant the theorem-proving approach. Rather, we would hope that techniques from each one can inspire advances in the other. For example, Levesque and his colleagues have been pursuing a course of trying to change the semantics of formulas in order to make it easier to check their validity [Levesque, 1984b; Lakemeyer, 1986; Patel-Schneider, 1985]. Levesque's approach involves considering possible *situations* rather than possible worlds. A situation is a generalized world, where a primitive formula can be true, false, both, or neither. Does the model-checking problem also simplify in this context? In general, it seems that it does not, but perhaps some interesting special cases do. This issue may be worth exploring further.

Acknowledgements

The ideas in this paper have been brewing for a while (a brief paragraph discussing some of these issues appeared in [Halpern, 1987]). Discussions with Hector Levesque and Yoram Moses helped us formulate these issues. We'd like to thank Ron Fagin, Adam Grove, Gerhard Lakemeyer, and Hector Levesque for useful comments on an earlier draft of this paper.

References

- [Bacchus, 1990] F. Bacchus. *Representing and reasoning with probabilistic knowledge*. MIT Press, 1990.
- [Brachman, 1985] R. Brachman. I lied about the trees (or, defaults and definitions in knowledge representation). *The AI Magazine*, 6(3):80–93, 1985.
- [Burch et al., 1990] J. R. Burch, E. M. Clarke, D. L. Dill, J. Hwang, and K. L. McMillan. Symbolic model checking: 10^{20} states and beyond. In *Proc. 5th IEEE Symp. on Logic in Computer Science*, pages 428–439, 1990.
- [Chandy and Misra, 1986] K. Chandy and J. Misra. How processes learn. *Distributed Computing*, 1(1):40–52, 1986.
- [Clarke and Grümberg, 1987] E. M. Clarke and O. Grümberg. Research on automatic verifica-
- tion and finite-state concurrent systems. In J. Traub et al., editors, *Annual Review of Computer Science*, Vol. 2, pages 269–289. Annual Reviews Inc., 1987.
- [Clarke et al., 1986] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems*, 8(2):244–263, 1986.
- [Dwork and Moses, 1990] C. Dwork and Y. Moses. Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation*, 88(2):156–186, 1990.
- [Emerson and Halpern, 1985] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30(1):1–24, 1985.
- [Emerson et al., 1989] E. A. Emerson, T. Sadler, and J. Srinivasan. Efficient temporal reasoning. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 166–178, 1989.
- [Etherington et al., 1989] D. Etherington, A. Borgida, R. J. Brachman, and H. Kautz. Vivid knowledge and tractable reasoning: preliminary report. In *Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1146–1152, 1989.
- [Fagin and Halpern, 1988] R. Fagin and J. Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988.
- [Fagin and Halpern, 1989] R. Fagin and J. Y. Halpern. Uncertainty, belief, and probability. In *Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1161–1167, 1989. An expanded version of this paper appears as IBM Research Report RJ 6191, April 1988.
- [Fagin et al., 1983] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. In *Proc. 2nd ACM Symp. on Principles of Database Systems*, pages 352–365, 1983.
- [Fagin et al., 1984] R. Fagin, J. Y. Halpern, and M. Y. Vardi. A model-theoretic analysis of knowledge: preliminary report. In *Proc. 25th IEEE Symp. on Foundations of Computer Science*, pages 268–278, 1984. A revised and expanded version appears as IBM Research Report RJ 6641, 1988, and will appear in *Journal of the ACM*.
- [Fagin et al., 1986] R. Fagin, J. Y. Halpern, and M. Y. Vardi. What can machines know? On the epistemic properties of machines. In *Proc. of National Conference on Artificial Intelligence (AAAI-86)*, pages 428–434, 1986. A revised and expanded version appears as IBM Research Report RJ 6250, 1988, under the name *What can machines know? On the properties of knowledge in distributed systems*, and will appear in *Journal of the ACM*.

- [Fikes and Nilsson, 1972] R. E. Fikes and N. J. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1972.
- [Gardenfors, 1988] P. Gardenfors. *Knowledge in Flux*. Cambridge University Press, 1988.
- [Halpern and Fagin, 1989] J. Y. Halpern and R. Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3(4):159–179, 1989.
- [Halpern and Moses, 1984] J. Y. Halpern and Y. Moses. Towards a theory of knowledge and ignorance. In *Proc. AAAI Workshop on Non-monotonic Logic*, pages 125–143, 1984. Reprinted in *Logics and Models of Concurrent Systems*, (ed. K. Apt), Springer-Verlag, 1985, pp. 459–476.
- [Halpern and Moses, 1990] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. An early version appeared in *Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing*, 1984.
- [Halpern and Zuck, 1987] J. Y. Halpern and L. D. Zuck. A little knowledge goes a long way: Simple knowledge-based derivations and correctness proofs for a family of protocols. In *Proc. 6th ACM Symp. on Principles of Distributed Computing*, pages 269–280, 1987. A revised and expanded version appears as IBM Research Report RJ 5857, 1987 and will appear in *Journal of the ACM*.
- [Halpern et al., 1988] J. Y. Halpern, Y. Moses, and M. R. Tuttle. A knowledge-based analysis of zero knowledge. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 132–147, 1988.
- [Halpern, 1986] J. Y. Halpern. Reasoning about knowledge: an overview. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference*, pages 1–17. Morgan Kaufmann, 1986. Reprinted in *Proc. National Computer Conference*, 1986, pp. 219–228.
- [Halpern, 1987] J. Y. Halpern. Using reasoning about knowledge to analyze distributed systems. In J. Traub et al., editors, *Annual Review of Computer Science*, Vol. 2, pages 37–68. Annual Reviews Inc., 1987.
- [Halpern, 1990] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.
- [Hamilton and Delgrande, 1989] S. J. Hamilton and J. P. Delgrande. An investigation of modal structures as an alternative semantic basis for epistemic logics. *Computational Intelligence*, 5:82–96, 1989.
- [Hintikka, 1975] J. Hintikka. Impossible possible worlds vindicated. *Journal of Philosophical Logic*, 4:475–484, 1975.
- [Holtzmann, 1988] G. J. Holtzmann. An improved protocol reachability analysis technique. *Software Practice and Experience*, 18(2):137–161, 1988.
- [Konolige, 1984] K. Konolige. Circumscriptive ignorance. In *Proc. of National Conference on Artificial Intelligence (AAAI-84)*, pages 202–204, 1984.
- [Lakemeyer, 1986] G. Lakemeyer. Steps towards a first-order logic of explicit and implicit belief. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference*, pages 325–340. Morgan Kaufmann, 1986.
- [Lejoly and Minsoul, 1990] Ph. Lejoly and M. Minsoul. A subjective logic of knowledge. Unpublished manuscript, 1990.
- [Levesque, 1981] H. J. Levesque. The interaction with incomplete knowledge bases: a formal treatment. In *Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, pages 240–245, 1981.
- [Levesque, 1984a] H. J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.
- [Levesque, 1984b] H. J. Levesque. A logic of implicit and explicit belief. In *Proc. of National Conference on Artificial Intelligence (AAAI-84)*, pages 198–202, 1984.
- [Levesque, 1986] H. J. Levesque. Making believers out of computers. *Artificial Intelligence*, 30:81–108, 1986.
- [Levesque, 1987] H. J. Levesque. All I know: an abridged report. In *Proc. of National Conference on Artificial Intelligence (AAAI-87)*, pages 426–431, 1987. A longer version under the title “All I know: a study in autoepistemic logic” appears as University of Toronto technical report KRR-TR-89-3, and will appear in *Artificial Intelligence*.
- [Lichtenstein and Pnueli, 1985] O. Lichtenstein and A. Pnueli. Checking the finite-state concurrent programs satisfy their linear specifications. In *Proc. 13th ACM Symp. on Principles of Programming Languages*, pages 97–107, 1985.
- [Manna and Pnueli, 1981] Z. Manna and A. Pnueli. Verification of temporal programs: the temporal framework. In R. S. Boyer and J. S. Moore, editors, *The Correctness Problem in Computer Science*. Academic Press, 1981.
- [McCarthy, 1968] J. McCarthy. Programs with common sense. In M. Minsky, editor, *Semantic Information Processing*, pages 403–418. The MIT Press, Cambridge, MA, 1968.
- [Moore, 1985] R. C. Moore. A formal theory of knowledge and action. In J. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex Publishing Corp., 1985.
- [Morris and Nado, 1986] P. H. Morris and R. A. Nado. Representing actions with an assumption-based

- truth maintenance system. In *Proc. of National Conference on Artificial Intelligence (AAAI-86)*, pages 13–17, 1986.
- [Moses and Tuttle, 1988] Y. Moses and M. R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3:121–169, 1988.
- [Moses, 1988] Y. Moses. Resource-bounded knowledge. In M. Y. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 261–276. Morgan Kaufmann, 1988.
- [Nilsson, 1986] N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.
- [Parikh and Ramanujam, 1985] R. Parikh and R. Ramanujam. Distributed processing and the logic of knowledge. In R. Parikh, editor, *Proc. of the Workshop on Logics of Programs*, pages 256–268, 1985.
- [Parikh, 1984] R. Parikh. Logics of knowledge, games, and dynamic logic. In *FST-TCS*, Lecture Notes in Computer Science, Vol. 181, pages 202–222. Springer-Verlag, 1984.
- [Patel-Schneider, 1985] P. F. Patel-Schneider. A decidable first-order logic for knowledge representation. In *Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 455–458, 1985.
- [Pearl and Verma, 1987] J. Pearl and T. Verma. The logic of representing dependencies by directed graphs. In *aaai87*, pages 374–379, 1987.
- [Queille and Sifakis, 1981] J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. In *Proc. 5th Int'l Symp. on Programming*, 1981.
- [Reiter, 1984] R. Reiter. Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On Conceptual Modelling*, pages 191–233. Springer-Verlag, 1984.
- [Rosenschein and Kaelbling, 1986] S. J. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference*, pages 83–97. Morgan Kaufmann, 1986.
- [Rosenschein, 1985] S. J. Rosenschein. Formal theories of AI in knowledge and robotics. *New Generation Computing*, 3:345–357, 1985.
- [Shoham, 1987] Y. Shoham. A semantical approach to nonmonotonic logics. In *Proc. 2nd IEEE Symp. on Logic in Computer Science*, pages 275–279, 1987.
- [Sistla and Clarke, 1985] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.
- [Subramanian and Genesereth, 1987] D. Subramanian and M. R. Genesereth. The relevance of irrelevance. In *ijcai87*, pages 416–422, 1987.
- [Vardi and Wolper, 1986] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st IEEE Symp. on Logic in Computer Science*, pages 332–344, 1986.
- [Vardi, 1982] M. Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 137–146, 1982.
- [Vardi, 1985] M. Y. Vardi. A model-theoretic analysis of monotonic knowledge. In *Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 509–512, 1985.
- [Vardi, 1986] M. Y. Vardi. Querying logical databases. *Journal of Computer and System Sciences*, 33:142–160, 1986.

Qualifying Number Restrictions in Concept Languages

Bernhard Hollunder and Franz Baader
 German Research Center for Artificial Intelligence (DFKI)
 Projectgroup WINO
 Postfach 2080, W-6750 Kaiserslautern, Germany
 E-mail: {hollunde, baader}@dfki.uni-kl.de

Abstract

We investigate the subsumption problem in logic-based knowledge representation languages of the KL-ONE family. The language presented in this paper provides the constructs for conjunction, disjunction, and negation of concepts, as well as qualifying number restrictions. The latter ones generalize the well-known role quantifications (such as value restrictions) and ordinary number restrictions, which are present in almost all KL-ONE based systems. Until now, only little attempts were made to integrate qualifying number restrictions into concept languages. It turns out that all known subsumption algorithms which try to handle these constructs are incomplete, and thus detecting only few subsumption relations between concepts. We present a subsumption algorithm for our language which is sound and complete. Subsequently we discuss why the subsumption problem in this language is rather hard from a computational point of view. This leads to an idea of how to recognize concepts which cause tractable problems.

1 INTRODUCTION

Knowledge representation systems of the KL-ONE family such as

- KL-ONE [Brachman and Schmolze, 1985],
- BACK [Nebel, 1990],
- CLASSIC [Borgida et al., 1989],
- KANDOR [Patel-Schneider, 1984],
- KL-TWO [Vilain, 1985],
- KRYPTON [Brachman et al., 1985],
- LOOM [MacGregor and Bates, 1987],

provide so-called *concept languages* for expressing taxonomical knowledge. Concept languages allow the definition of *concepts* which are built out of primitive concepts and roles. The primitive concepts are interpreted as sets of individuals and the roles are interpreted as binary relations between individuals. Starting with primitive concepts and roles one can build concepts using various language constructs. Concepts are again interpreted as sets of individuals.

To give an example, assume that *person*, *male*, and *shy* are primitive concepts, and *child* is a role. If constructs such as conjunction, disjunction, and negation of concepts are available in the concept language, one can express “persons that are male or not shy” by $\text{person} \sqcap (\text{male} \sqcup \neg\text{shy})$. Since concepts are interpreted as sets, conjunction of concepts (\sqcap) can be interpreted as set intersection, disjunction of concepts (\sqcup) as set union, and negation of concepts (\neg) as set complement.

Qualifying number restrictions provide restrictions on roles which for instance allow to describe “individuals with at least two children who are shy” by

$$(\geq 2 \text{ child shy}),$$

and “individuals with at most three children who are male and not shy” by

$$(\leq 3 \text{ child } (\text{male} \sqcap \neg\text{shy})).$$

It turns out that the well-known role quantifications and ordinary number restrictions, which are available in almost all concept languages, are special cases of qualifying number restrictions.

Role quantifications are of the form $\exists \text{child}.\text{male}$ and $\forall \text{child}.\text{shy}$ (or (SOME child male) and (ALL child shy) in a Lisp-like notation). These expressions can be read as “individuals having (at least) one male child” and “individuals for whom all children are shy”, respectively. Obviously, $\exists \text{child}.\text{male}$ means the same as $(\geq 1 \text{ child male})$, and $\forall \text{child}.\text{shy}$ means the same as $(\leq 0 \text{ child } \neg\text{shy})$.

Ordinary number restrictions are of the form $(\geq n R T)$ and $(\leq n R T)$, where the special concept symbol *T* denotes the set of all individuals (of the

interpretation). Qualifying number restrictions generalize these ordinary number restrictions as follows: a specified number of role fillers for a role can be restricted to arbitrary concepts rather than only to T .

The main reasoning facilities concerning the taxonomical knowledge in the above mentioned systems are the

- determination whether a concept is *satisfiable*, i.e., whether a concept denotes a nonempty set in some interpretation, and the
- computation of the *subsumption* relation between concepts.

A concept C *subsumes* (is more general than) a concept D iff in every interpretation the set denoted by C is a superset of the set denoted by D . For example,

$$(\leq 1 \text{ child } (\text{male} \sqcap \neg \text{shy}))$$

subsumes

$$(\geq 2 \text{ child } (\text{male} \sqcap \text{shy})) \sqcap (\leq 3 \text{ child male}),$$

since every individual with at least two male and shy children and at most three male children has at most one child which is male and not shy. This example demonstrates that, in contrast to other concept languages, already for small concepts it is not that apparent whether there exists a subsumption relation between them.

The subsumption problem can be reduced to the satisfiability problem if the concept language provides conjunction and negation of concepts. In fact, a concept C subsumes a concept D if and only if the concept $\neg C \sqcap D$ is not satisfiable. Thus, an algorithm which checks satisfiability of concepts also yields an algorithm which checks subsumption between concepts.

If one considers the subsumption algorithms used in the above mentioned KL-ONE systems, then it turns out that these algorithms are *incomplete*. A reason for this fact is that until recently only incomplete subsumption algorithms were known for non-trivial concept languages. An incomplete subsumption algorithm has the property that it sometimes fails to recognize that a concept subsumes another one. [Schmidt-Schaub and Smolka, 1988] were the first who gave a sound and complete subsumption algorithm for the concept language ALC which provides conjunction, disjunction, and negation of concepts, as well as role quantifications. Actually, they presented an algorithm for checking satisfiability of concepts. Given a concept C the algorithm tries to generate a finite interpretation such that the set denoted by C is nonempty. If this process fails, i.e., if a contradiction occurs, such an interpretation cannot exist and C is not satisfiable, otherwise C is satisfiable.

[Hollunder, Nutt, and Schmidt-Schaub, 1990] demonstrate that the ideas underlying the algorithm which

checks satisfiability for the language ALC can be applied to languages using also other constructs such as ordinary number restrictions and conjunction of roles. Moreover, they argue that these ideas yield a general methodology for devising satisfiability (and hence subsumption) algorithms for concept languages.

The present paper pursues several goals. Firstly, it describes a sound and complete algorithm for checking satisfiability for a very expressive concept language containing not only conjunction, disjunction, and negation of concepts, but also qualifying number restrictions. Qualifying number restrictions are not just artificial constructs, but they are partly available in some systems. For example, the concept language used in KANDOR allows qualifying number restrictions in the restricted form $(\geq n R C)$ and $(\leq n R T)$. Recently, qualifying number restrictions have been included into the assertional part ("A-Box") of the system MESON [Owsinicki-Klewe, 1990].

Secondly, the paper exemplifies the claim that there is in fact a general methodology for devising satisfiability algorithms for concept languages. However, devising a satisfiability algorithm for concept languages with qualifying number restrictions is of a rather different quality. The reason is that qualifying number restrictions allow to generate inherently complex term structures causing subtle combinatorial problems.

Finally, the paper investigates what kind of language constructs, and in what kind of combination may lead to long computations of the presented algorithm. This leads to an idea of how to single out concepts which can be checked fast on satisfiability.

The paper is organized as follows. In the next section we formally introduce the syntax and semantics of our concept language. In Section 3 we present a calculus for deciding satisfiability of concepts of this language. A functional algorithm which is extracted from the calculus is given in Section 4. In Section 5 we give ideas of how to recognize concepts which cause tractable problems. Finally, in Section 6 we discuss the claim that there is a general methodology for devising satisfiability (and hence subsumption) algorithms for concept languages.

2 THE LANGUAGE

We assume two disjoint alphabets of symbols, called *primitive concepts* and *roles*. We usually denote primitive concepts by the letter A and roles by the letter R . The special primitive concepts T and \perp are called *top* and *bottom*. The set of *concepts* is inductively defined as follows. Every primitive concept is a concept. Now let C and D be concepts already defined, let R be a

role, and let n be a nonnegative integer. Then

$C \sqcap D$	(conjunction)
$C \sqcup D$	(disjunction)
$\neg C$	(negation)
$(\geq n R C)$	(at-least restriction)
$(\leq n R C)$	(at-most restriction)

are concepts. The at-least and at-most restrictions are also called *qualifying number restrictions*.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ of our language consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of \mathcal{I}) and a function $.^{\mathcal{I}}$ (the *interpretation function* of \mathcal{I}). The interpretation function maps every primitive concept A to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and every role R to a subset $R^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The special concepts T and \perp are interpreted as $\Delta^{\mathcal{I}}$ and the empty set, respectively. The interpretation function—which gives an interpretation for primitive concepts and roles—can be extended to arbitrary concepts as follows. Let C and D be concepts, let R be a role, and let n be a nonnegative integer. Assume that $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$ are already defined. Then

$$\begin{aligned}(C \sqcap D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (C \sqcup D)^{\mathcal{I}} &:= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\ (\neg C)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (\geq n R C)^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid |aR^{\mathcal{I}} \cap C^{\mathcal{I}}| \geq n\} \\ (\leq n R C)^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid |aR^{\mathcal{I}} \cap C^{\mathcal{I}}| \leq n\}\end{aligned}$$

where $aR^{\mathcal{I}} := \{b \in \Delta^{\mathcal{I}} \mid (a, b) \in R^{\mathcal{I}}\}$, and $|X|$ denotes the cardinality of the set X .

Our language allows to express the well-known *role quantifications* (see e.g. [Nebel and Smolka, 1990]), which are present in almost all concept languages, as follows:

$$\begin{aligned}\forall R.C &:= (\leq 0 R \neg C) \\ \exists R.C &:= (\geq 1 R C).\end{aligned}$$

The construct $\forall R.C$ is usually called *value restriction*, and the construct $\exists R.C$ is e.g. called *c-some* in [Nebel, 1990]. Note that the concept $(\leq 0 R \neg C)$ denotes the set of all elements for which every role filler for R is in C . This shows that the definition for the construct $\forall R.C$ we have given above coincides with the semantics usually given for this construct.

Now we are able to define the language *ALC* of [Schmidt-Schaub and Smolka, 1988] as sublanguage of our language. This concept language allows conjunction, disjunction, and negation of concepts, as well as role quantifications.

As already mentioned in the introduction, a concept C is called *satisfiable* iff there exists an interpretation \mathcal{I} such that $C^{\mathcal{I}}$ is nonempty. We say C *subsumes* D iff $C^{\mathcal{I}} \supseteq D^{\mathcal{I}}$ for every interpretation \mathcal{I} , and C is *equivalent* to D iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every interpretation \mathcal{I} .

Since our language allows conjunction and negation of concepts, satisfiability and subsumption of concepts can be reduced in linear time to each other.

Proposition 2.1 *Let C and D be concepts. Then:*

1. *C subsumes D if and only if $\neg C \sqcap D$ is not satisfiable.*
2. *C is satisfiable if and only if \perp does not subsume C .*

Thus an algorithm for checking satisfiability of concepts can also be used for deciding subsumption of concepts and vice versa.

In the following section we present an algorithm that decides satisfiability of concepts. To keep this algorithm simple it is convenient to transform concepts into normal forms. We say a concept C is in *negation normal form* if negation signs in C appear only immediately in front of primitive concepts different from T and \perp . Negation normal forms can be computed using the following *simplification rules*:

$$\begin{aligned}\neg T &\longrightarrow \perp \\ \neg \perp &\longrightarrow T \\ \neg(C \sqcap D) &\longrightarrow \neg C \sqcup \neg D \\ \neg(C \sqcup D) &\longrightarrow \neg C \sqcap \neg D \\ \neg\neg C &\longrightarrow C \\ \neg(\leq n R C) &\longrightarrow (\geq (n+1) R C) \\ \neg(\geq n R C) &\longrightarrow \begin{cases} \perp & \text{if } n = 0 \\ (\leq (n-1) R C) & \text{if } n > 0. \end{cases}\end{aligned}$$

Let C be a concept. By $NNF(C)$ we denote the concept which is obtained from C by applying the simplification rules in top-down, left to right order as long as possible. For example,

$$NNF(\neg(A \sqcup (\geq 3 R B))) = (\neg A \sqcap (\leq 2 R B)).$$

The following result can be proved easily.

Proposition 2.2 *Let C be a concept. Then $NNF(C)$*

1. *is in negation normal form,*
2. *is equivalent to C , and*
3. *can be computed in linear time.*

3 CHECKING SATISFIABILITY

In this section we describe an algorithm for deciding satisfiability in our concept language. The algorithm uses so-called constraints, which are built out of variables, concepts, and roles. Sets of constraints will be modified with the help of completion rules. The notions constraints and completion rules have already been used in [Schmidt-Schaub and Smolka, 1988, Hollunder, Nutt, and Schmidt-Schaub, 1990].

We assume that there exists an alphabet of variable symbols, which will be denoted by the letters $x, y, z,$

and u . A *constraint* has one of the following forms

$$x : C, \quad xRy,$$

where C is a concept in negation normal form and R is a role. For xRy we say that y is an *R -successor* of x . If R is clear from the context or irrelevant, we simply say that y is a *successor* of x .

The interpretation of constraints is defined as follows. Let \mathcal{I} be an interpretation of the concept language. An \mathcal{I} -*assignment* is a function α that maps every variable to an element of $\Delta^{\mathcal{I}}$. We say that α *satisfies* $x : C$ iff

$$\alpha(x) \in C^{\mathcal{I}},$$

and α *satisfies* xRy iff

$$(\alpha(x), \alpha(y)) \in R^{\mathcal{I}}.$$

A constraint s is *satisfiable* iff there exists an interpretation \mathcal{I} and an \mathcal{I} -assignment α such that α satisfies s . A *constraint system* S is a finite, nonempty set of constraints. An \mathcal{I} -assignment α *satisfies* a constraint system S iff α satisfies every constraint in S . A constraint system S is *satisfiable* iff there exists an interpretation \mathcal{I} and an \mathcal{I} -assignment α such that α satisfies S .

Proposition 3.1 *A concept C in negation normal form is satisfiable if and only if the constraint system $\{x : C\}$ is satisfiable.*

Proof. Follows immediately from the definitions. \square

Given a concept C in negation normal form which has to be checked for satisfiability, our calculus starts with the constraint system $\{x : C\}$. Then, by applications of completion rules, it adds constraints to this system until a contradiction occurs, or an interpretation \mathcal{I} such that $C^{\mathcal{I}}$ is nonempty can be immediately obtained from the actual constraint system.

Before we formulate the rules we need some more definitions.

- For a variable x in a constraint system S we want to count the R -successors which are in a certain concept C . We therefore define

$$n_{R,C,S}(x) := |\{y \mid \{xRy, y : C\} \subseteq S\}|.$$

- By $[y/z]S$ we denote the constraint system that is obtained from S by replacing each occurrence of y by z .

We have the following rules:

1. $S \rightarrow_n \{x : C_1, x : C_2\} \cup S$
if $x : C_1 \sqcap C_2$ is in S ,
 $x : C_1$ and $x : C_2$ are not both in S
2. $S \rightarrow_u \{x : D\} \cup S$
if $x : C_1 \sqcup C_2$ is in S ,
neither $x : C_1$ nor $x : C_2$ is in S ,
and $D = C_1$ or $D = C_2$

3. $S \rightarrow_{choose} \{y : C^*\} \cup S$

if $x : (\leq n R C)$ is in S ,
 xRy is in S ,
neither $y : C$ nor $y : NNF(\neg C)$ is in S ,
and $C^* = C$ or $C^* = NNF(\neg C)$

4. $S \rightarrow_{\geq} \{xRy, y : C\} \cup S$

if $x : (\geq n R C)$ is in S , $n_{R,C,S}(x) < n$,
and y is a new variable

5. $S \rightarrow_{\leq} [y/z]S$

if $x : (\leq n R C)$ is in S ,
 $xRy, y : C, xRz, z : C$ are in S , $y \neq z$,
 $n_{R,C,S}(x) > n$,
for every u with xRu
either $u : C$ or $u : NNF(\neg C)$ is in S .

Note that to every language construct—except negation of concepts—there corresponds a rule. Since concepts are assumed to be in negation normal form, we do not need rules which handle concepts with negation as outermost symbol such as $x : \neg(C \sqcap D)$, $x : \neg(C \sqcup D)$ etc. Negation applied to primitive concepts will be treated by the definition of “clash” given below.

The rules are used to decide whether a given constraint system is satisfiable as follows. We apply the rules until we obtain a constraint system such that no rule is applicable to it (a so-called complete constraint system). For such a constraint system it is easy to decide whether it is satisfiable or not.

Now let us discuss the rules. Obviously, the \rightarrow_n - and \rightarrow_u -rule are decomposing constraints having the form $x : C \sqcap D$ and $x : C \sqcup D$, respectively. Both rules are defined as in [Schmidt-Schauf and Smolka, 1988]. The rules which treat qualifying number restrictions are similar to the rules for ordinary number restrictions used in [Hollunder, Nutt, and Schmidt-Schauf, 1990]. However, in the case of ordinary number restrictions one only has to know for a given variable x all its R -successors for some role R ; whereas in the case of qualifying number restrictions one has to know for a given variable x all its R -successors which are in a certain concept C . This fact leads to the definition of $n_{R,C,S}(x)$ given above. However, the number $n_{R,C,S}(x)$ is only useful if the \rightarrow_{choose} -rule does not apply to constraints containing the variable x . To see this assume that S is a constraint system such that $x : (\leq n R C)$ and xRy are in S , but neither $y : C$ nor $y : NNF(\neg C)$ is in S . Then $n_{R,C,S}(x)$ would not count the variable y since $y : C$ is not in S . Nevertheless there may exist an interpretation \mathcal{I} and an \mathcal{I} -assignment α such that $\alpha(y) \in C^{\mathcal{I}}$. That means that in this interpretation x may have more R -successors in C than are indicated by $n_{R,C,S}(x)$, i.e., we may have

$$n_{R,C,S}(x) < |\{\alpha(y) \mid (\alpha(x), \alpha(y)) \in R^{\mathcal{I}} \wedge \alpha(y) \in C^{\mathcal{I}}\}|.$$

Thus $n \geq n_{R,C,S}(x)$ need not imply

$$n \geq |\{\alpha(y) \mid (\alpha(x), \alpha(y)) \in R^{\mathcal{I}} \wedge \alpha(y) \in C^{\mathcal{I}}\}|.$$

This means that the constraint $x : (\leq n R C)$ can be violated in spite of the fact that $n \geq n_{R,C,S}(x)$. To overcome this problem we use the \rightarrow_{choose} -rule which is based on the following idea. We know that for every I and α we have $\alpha(y) \in C^I$ or $\alpha(y) \in (\neg C)^I$. An application of the \rightarrow_{choose} -rule nondeterministically adds the correct constraint $y : C$ or $y : NNF(\neg C)$ to S .

Now suppose that for every R -successor u of x either $u : C$ or $u : NNF(\neg C)$ is in S . If $x : (\leq n R C)$ is in S and $n_{R,C,S}(x) > n$, then there are too many R -successors of x in C . An application of the \rightarrow_{\leq} -rule reduces their number by identifying two previously different R -successors.

For constraints of the form $x : (\geq n R C)$ it is not necessary to apply the \rightarrow_{choose} -rule to all successors y of x . In fact, assume that xRy is in S , but neither $y : C$ nor $y : NNF(\neg C)$ is in S . In this case it does not lead to problems if the \rightarrow_{\geq} -rule introduces a new variable because y was not counted for $n_{R,C,S}(x)$. If I is an interpretation and α is an I -assignment such that $\alpha(y) \in C^I$, we may now have

$$n_{R,C,S}(x) < |\{\alpha(y) \mid (\alpha(x), \alpha(y)) \in R^I \wedge \alpha(y) \in C^I\}|.$$

But then $n \leq n_{R,C,S}(x)$ also implies

$$n \leq |\{\alpha(y) \mid (\alpha(x), \alpha(y)) \in R^I \wedge \alpha(y) \in C^I\}|.$$

If $x : (\geq n R C)$ is in S and $n_{R,C,S}(x) < n$, then there may not be enough R -successors of x in C . An application of the \rightarrow_{\geq} -rule generates a new variable y and adds the constraints xRy and $y : C$ to S . Thus, the number $n_{R,C,S}(x)$ is increased by one.

The rules as described above have the disadvantage that they allow infinite chains of rule applications.

Example 3.2 Consider the constraint system

$$S = \{x : (\geq 2 R A) \sqcap (\leq 1 R A)\}.$$

We obtain the following constraint systems by applications of the rules:

$$\begin{aligned} S &\xrightarrow{\rightarrow_n} S \cup \{x : (\geq 2 R A), x : (\leq 1 R A)\} =: S_1 \\ &\xrightarrow{\rightarrow_{\geq}} S_1 \cup \{xRy, y : A\} =: S_2 \\ &\xrightarrow{\rightarrow_{\geq}} S_2 \cup \{xRz, z : A\} =: S_3 \\ &\xrightarrow{\rightarrow_{\leq}} [z/y]S_3 = S_2. \end{aligned}$$

Thus we have the infinite chain $S \xrightarrow{\rightarrow_n} S_1 \xrightarrow{\rightarrow_{\geq}} S_2 \xrightarrow{\rightarrow_{\geq}} S_3 \xrightarrow{\rightarrow_{\leq}} S_2 \xrightarrow{\rightarrow_{\geq}} S_3 \dots$

The example demonstrates that alternating applications of the \rightarrow_{\geq} - and \rightarrow_{\leq} -rule are responsible for infinite chains. The idea to avoid such infinite chains is as follows. We only apply the \rightarrow_{\leq} -rule to a constraint system S (yielding the constraint system S') if every constraint of the form $x : (\geq n R C)$ that is satisfied in S , i.e., $n_{R,C,S}(x) \geq n$, remains satisfied in S' . Note that this condition is violated in the above example since $S_3 \xrightarrow{\rightarrow_{\leq}} S_2$, $n_{R,A,S_3}(x) = 2$ and $n_{R,A,S_2}(x) = 1$.

Before we can formulate the modified \rightarrow_{\leq} -rule we need the following definition. Let S be a constraint system. We say that the replacement of y by z is *safe* in S if for all x, R, C, n

$$\{xRy, y : C, xRz, z : C, x : (\geq n R C)\} \subseteq S$$

implies $n_{R,C,S}(x) > n$.

We reformulate the \rightarrow_{\leq} -rule as follows. Let S be a constraint system.

$$5^*. S \xrightarrow{\rightarrow_{\leq}} [y/z]S$$

if $x : (\leq n R C)$ is in S ,
 $xRy, y : C, xRz, z : C$ are in S , $y \neq z$,
 $n_{R,C,S}(x) > n$,
for every u with xRu
either $u : C$ or $u : NNF(\neg C)$ is in S ,
and the replacement of y by z is safe in S .

We will see that the additional condition ensures that there is no infinite chain of rule applications.

The *completion rules* consist of the \rightarrow_n -, \rightarrow_u -, \rightarrow_{choose} -, \rightarrow_{\geq} -, and \rightarrow_{\leq} -rule.

The following proposition, which one can prove easily, states that the completion rules are sound.

Proposition 3.3 *Let S and S' be constraint systems. Then:*

1. *If S' is obtained from S by application of the (deterministic) \rightarrow_n - or \rightarrow_{\geq} -rule, then S is satisfiable if and only if S' is satisfiable.*
2. *If S' is obtained from S by application of the (non-deterministic) \rightarrow_u -, \rightarrow_{choose} -, or \rightarrow_{\leq} -rule, then S is satisfiable if S' is satisfiable. Furthermore, if a nondeterministic rule applies to S , then it can be applied in such a way that it yields a constraint system S' such that S' is satisfiable if and only if S is satisfiable.*

The proof of termination of the completion rules—which is less obvious—will employ techniques which have been developed for proving termination of term rewriting systems (see [Dershowitz and Manna, 1979]).

Proposition 3.4 *Let C_0 be a concept in negation normal form. Then there is no infinite chain of applications of completion rules issuing from $\{x_0 : C_0\}$.*

A constraint system S which can be obtained from $\{x_0 : C_0\}$ by a finite number of applications of completion rules will be called *derived system* in the following. In order to prove the proposition, any derived system S will be mapped on an element $\Psi(S)$ of a set Q which is equipped with a well-founded strict partial ordering \gg . Since the ordering is well-founded, i.e., has no infinitely decreasing chains, termination will follow immediately as soon as one has established

the following property. Whenever S' is obtained from the derived system S by application of a rule, one has $\Psi(S) \gg \Psi(S')$.

The elements of the set Q will have a rather complex structure. They are finite multisets of 5-tuples. Each component of the tuples is either a finite multiset of nonnegative integers (for the second, third, and fifth component) or a nonnegative integer (for the first and fourth component). Multisets are like sets, but allow multiple occurrences of identical elements. For example, $\{2, 2, 2\}$ is a multiset which is distinct from the multiset $\{2\}$. A given ordering on a set T can be extended to form an ordering on the finite multisets over T . In this ordering, a finite multiset M is larger than a finite multiset M' iff M' can be obtained from M by replacing one or more elements in M by any finite number of elements taken from T , each of which is smaller than one of the replaced elements. For example, $\{2, 2, 2\}$ is larger than $\{2\}$ and $\{2, 2, 1, 1, 0\}$. [Dershowitz and Manna, 1979] show that the induced ordering on finite multisets over T is well-founded if the original ordering on T is so.

The nonnegative integer components of our 5-tuples are compared with respect to the usual ordering on integers, and the finite multiset components by the multiset ordering induced by this ordering. The whole tuples are ordered lexicographically from left to right, i.e., (c_1, \dots, c_5) is larger than (c'_1, \dots, c'_5) iff there exists $i, 1 \leq i \leq 5$, such that $c_1 = c'_1, \dots, c_{i-1} = c'_{i-1}$, and c_i is larger than c'_i . Since the orderings on the components are well-founded, the lexicographical ordering on the tuples is also well-founded. Finite multisets of these tuples are now compared with respect to the multiset ordering induced by this lexicographical ordering. This is the well-founded ordering \gg on Q mentioned above.

Before we can define the mapping Ψ from derived systems to elements of Q , we need two more definitions. For two nonnegative integers n, m we denote by $n \dot{-} m$ the asymmetrical difference between n and m , i.e., $n \dot{-} m := n - m$ if $n \geq m$, and $n \dot{-} m := 0$ if $n < m$. For a concept C the size $|C|$ is inductively defined as

- $|A| = 1$ for all primitive concepts A ,
- $|\neg C| = |C|$,
- $|(\geq n R C)| = |(\leq n R C)| = 1 + |C|$,
- $|C \sqcap D| = |C \sqcup D| = |C| + |D|$.

Definition 3.5 Let S be a constraint system. Then $\Psi(S)$ is the multiset which contains for each variable x occurring in S the following 5-tuple $\psi(x)$:

1. The first component of $\psi(x)$ is the nonnegative integer $\max\{|C| \mid x : C \text{ is in } S\}$.
2. The second component of $\psi(x)$ is a multiset which contains, for each constraint $x : C \sqcap D$ (resp.

$x : C \sqcup D$) in S for which the \rightarrow_{\sqcap} -rule (resp. \rightarrow_{\sqcup} -rule) is applicable, the nonnegative integer $|C \sqcap D|$ (resp. $|C \sqcup D|$).

3. The third component of $\psi(x)$ is a multiset which contains, for each constraint $x : (\geq n R C)$, the nonnegative integer $n \dot{-} n_{R,C,S}(x)$.
4. The fourth component of $\psi(x)$ is the number of all successors of x in S .
5. The fifth component of $\psi(x)$ is a multiset which contains, for each constraint $x : (\leq n R C)$ in S the number of all R -successors y of x such that neither $y : C$ nor $y : \text{NNF}(\neg C)$ is in S .

For the constraint systems of Example 3.2 we have

$$\begin{aligned}\Psi(S) &= \{(4, \{4\}, \emptyset, \emptyset, \emptyset)\} \\ \Psi(S_1) &= \{(4, \emptyset, \{2\}, 0, \{0\})\} \\ \Psi(S_2) &= \{(4, \emptyset, \{1\}, 1, \{0\}), (1, \emptyset, \emptyset, 0, \emptyset)\} \\ \Psi(S_3) &= \{(4, \emptyset, \{0\}, 2, \{0\}), (1, \emptyset, \emptyset, 0, \emptyset), \\ &\quad (1, \emptyset, \emptyset, 0, \emptyset)\}\end{aligned}$$

Proposition 3.4 is now an immediate consequence of the next lemma.

Lemma 3.6 If S' is obtained from the derived system S by application of a completion rule then $\Psi(S) \gg \Psi(S')$.

The proof is relatively straightforward since the ordering on Q is defined in an appropriate way. It can be found in [Hollunder and Baader, 1991].

A constraint system is called *complete* if no completion rule applies to it. A constraint system S contains a *clash* iff

- $\{x : \perp\} \subseteq S$ for some variable x , or
- $\{x : A, x : \neg A\} \subseteq S$ for some variable x and some primitive concept A , or
- $\{x : (\leq n R C)\} \subseteq S$ and $n_{R,C,S}(x) > n$ for some variable x , role R , concept C , and nonnegative integer n , and the \rightarrow_{\leq} -rule is not applicable to $x : (\leq n R C)$.

The following proposition states that it is sufficient to concentrate on these kinds of contradictions in complete constraint systems.

Proposition 3.7 A complete constraint system is satisfiable if and only if it contains no clash.

Proof. Obviously, a complete constraint system containing a clash is not satisfiable.

To see the other direction assume that S is a clash free complete constraint system. We construct an interpretation \mathcal{I} and an \mathcal{I} -assignment α which satisfies

S. The domain $\Delta^{\mathcal{I}}$ of \mathcal{I} consists of all variables occurring in S . For all primitive concepts A different from \top and \perp we define $A^{\mathcal{I}} := \{x \mid x : A \in S\}$, and for all roles R we define $R^{\mathcal{I}} := \{(x, y) \mid x R y \in S\}$. The \mathcal{I} -assignment α is defined by mapping variables to themselves, i.e. $\alpha(x) := x$. We now prove that α satisfies every constraint s in S . If s has the form $x R y$, then α satisfies $x R y$ by definition of \mathcal{I} and α . If s has the form $x : C$, we show by *induction on the structure of C* that $\alpha(x) \in C^{\mathcal{I}}$.

Base case: If C is a primitive concept different from \top and \perp , then $\alpha(x) \in A^{\mathcal{I}}$ by definition of \mathcal{I} and α . If $C = \top$, then obviously $\alpha(x) \in \top^{\mathcal{I}}$. Since S is clash free we have $C \neq \perp$.

Induction step: If $C = \neg A$ for a primitive concept A , the constraint $x : A$ is not in S since S is clash free. Then $\alpha(x) \notin A^{\mathcal{I}}$ and $\alpha(x) \in \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$. Hence $\alpha(x) \in (\neg A)^{\mathcal{I}}$.

If $C = C_1 \sqcap C_2$, then the constraints $x : C_1$ and $x : C_2$ are in S since S is complete. By the induction hypothesis we know that $\alpha(x) \in C_1^{\mathcal{I}}$ and $\alpha(x) \in C_2^{\mathcal{I}}$, which implies $\alpha(x) \in C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$, and hence $\alpha(x) \in (C_1 \sqcap C_2)^{\mathcal{I}}$. Similarly, it can be shown that constraints of the form $x : C_1 \sqcup C_2$ are satisfied by \mathcal{I} and α .

Suppose $C = (\leq n R D)$. Since the \rightarrow_{choose} -rule is not applicable to S , for every y with $x R y$ either $y : D$ or $y : NNF(\neg D)$ is in S . By the induction hypothesis we know that $\alpha(y) \in D^{\mathcal{I}}$ if $y : D$ is in S , or $\alpha(y) \in (\neg D)^{\mathcal{I}}$ if $y : NNF(\neg D)$ is in S . Since S is complete and does not contain a clash, we have $n_{R,D,S}(x) \leq n$. Hence $\alpha(x) \in (\leq n R D)^{\mathcal{I}}$. Similarly, it can be shown that constraints of the form $x : (\geq n R D)$ are satisfied by \mathcal{I} and α . \square

Now we can formulate and prove the main result of this section.

Theorem 3.8 *Satisfiability and subsumption of concepts are decidable.*

Proof. We obtain a decision procedure for the satisfiability problem of concepts as follows. Let C be a concept. First we transform C into its negation normal form C' , which can be done in linear time. Then we generate the finitely many constraint systems issuing from $\{x : C'\}$. If one of these constraint systems is clash free, then C' (and hence C) is satisfiable; otherwise C is not satisfiable.

Since our concept language allows negation of concepts the subsumption problem can be reduced to the satisfiability problem. \square

Our concept language generalizes the concept language \mathcal{ALC} in which deciding satisfiability and subsumption of concepts are PSPACE-complete problems [Schmidt-Schaub and Smolka, 1988]. Thus we have

Proposition 3.9 *Satisfiability and subsumption of concepts are PSPACE-hard problems.*

4 IMPLEMENTATION

The algorithm for deciding satisfiability of concepts given in the previous section has the nice property that soundness and completeness can be proved in a relatively simple manner. However, the rule-based algorithm as described is not convenient for implementation purposes. The reasons will be discussed in the first part of this section. One can obtain an algorithm which is more suitable for an actual implementation by imposing an appropriate control on the applications of completion rules. Based on the idea of [Schmidt-Schaub and Smolka, 1988] who gave an optimized algorithm for the language \mathcal{ALC} we will present in the second part of this section an optimized algorithm for our language.

The following considerations show how the above presented algorithm may be optimized.

(1) Contradictions may occur early in the completion process. If a constraint system S contains clashes of the form $\{x : \perp\}$ or $\{x : A, x : \neg A\}$, then every complete constraint system extending S also contains a clash. Thus, the completion process can be stopped since S is not satisfiable.

(2) In general, constraint systems contain redundant constraints. For example, suppose that a constraint system S contains $x : C \sqcap D$. If $x : C$ and $x : D$ are also in S , then obviously no completion rule will apply to $x : C \sqcap D$ in the further completion process. Hence it is not necessary to keep $x : C \sqcap D$ in S if $x : C$ and $x : D$ are in S . Similar arguments hold for $x : C \sqcup D$. Constraints of the form $x : (\geq n R C)$ or $x : (\leq n R C)$ can be removed from S under the following circumstances:

- No constraints having the form $x : C \sqcap D$ or $x : C \sqcup D$ are in S .
- The \rightarrow_{choose} , \rightarrow_{\geq} , and \rightarrow_{\leq} -rules do not apply to $x : (\geq n R C)$ or $x : (\leq n R C)$. That means, every successor of x in the final complete system is already contained in S .

Both conditions guarantee that all constraints which are implicitly imposed on x by the system S are already explicitly present in S . Since all variables which are introduced in the further completion process are new and hence different from x , we will not get new unsatisfied constraints on x later on.

(3) Only small portions of a constraint system need to be kept in memory at a time. The idea for this comes from the following observation. Suppose x is a variable in S and no completion rule applies to constraints of the form $x : C$. Consequently, there will be no more

replacements between successors of x . Suppose y and z are different successors of x . Since neither y will be replaced by z nor z will be replaced by y in the further completion process, there does not exist a variable u in the final constraint system such that u can be reached both from y and z by role chains. This means that we can inspect the constraints which will be introduced by restrictions imposed on y independently from those introduced by restrictions imposed on z without loosing possible clashes. This is so because clashes are defined w.r.t. a single variable.

(4) To detect clashes of the form $x : (\leq n R C)$ in S , where $n_{R,C,S}(x) > n$, we have to consider only the R -successors of x . Thus an actual implementation may store for a variable x only its R -successors at a time. Furthermore, there is no need for explicitly storing constraints of the form xRy .

Taking these optimizations into account we will formulate a functional algorithm that decides the satisfiability of concepts. This algorithm is obtained from our rule-based calculus by imposing some control on the application of the completion rules.

The algorithm described in Figure 1 uses the functions *or* and *find*. We assume that the binary function *or* behaves as the LISP-or, i.e., the second argument is evaluated if and only if the first argument does not evaluate to true. Let L be a list and let f be a function. The call *find l in L such that f(l)* iteratively takes an element l of L until the function f applied to l evaluates to true. In this case *find* immediately returns true. Otherwise, if there does no exist an element l in L such that f applied to l evaluates to true, *find* returns false.

Let C be a concept in negation normal form. Suppose the function *satisfiable* in Figure 1 is called with arguments x and $S = \{x : C\}$. Then *satisfiable* proceeds as follows. First, it checks whether S contains a clash. This check is restricted to constraints containing the variable x . If S is clash free, the \rightarrow_{\sqcap} - and \rightarrow_{\sqcup} -rules are applied until all constraints of the form $x : C \sqcap D$ or $x : C \sqcup D$ are decomposed. If no at-least restriction is imposed on x , i.e., there is no constraint $x : (\geq n R C)$ with $n > 0$ in S , we are done, and *satisfiable* returns true. (Note that in this case every at-most restriction imposed on x is trivially satisfied since we have not yet introduced successors of x .) Otherwise we generate for every role R which occurs in a constraint $x : (\geq n R C)$ with $n > 0$ R -successors for x . This is done by the call *generate-successors(x, R, S)*. Note that successors of x which are related by different roles to x can be inspected independently from each other without loosing possible clashes.

Suppose that *generate-successors* is called with arguments x , R , and S . Then R -successors for the variable x are generated, which will be constrained according to the constraints imposed on x . The functions *satisfi-*

able and *generate-successors* are defined in such a way that every variable occurring in S which is not equal to x is an R -successor of x . Suppose $x : (\leq n R C)$ is in S . If the number of R -successors of x is not greater than n , then $x : (\leq n R C)$ is obviously satisfied in S . Thus, it is checked whether $|\{z \mid z \text{ is a variable in } S, z \neq x\}|$ (i.e., the number of R -successors of x) is greater than n . If this is the case, it is tested whether every R -successor of x is either in C or in $NNF(\neg C)$. Suppose y is a variable in S not equal to x . Thus y is an R -successor of x . Suppose further that neither $y : C$ nor $y : NNF(\neg C)$ is in S . If $x : (\leq 0 R C)$ is in S , we add $y : NNF(\neg C)$ to S . Obviously, adding $y : C$ to S would lead to a clash. Otherwise, if $n > 0$, we cannot exclude one of the two possibilities. We only know that y is either in C or in $\neg C$. This nondeterminism is solved by generating two recursive calls of *generate-successors* covering both cases.

Next it is checked whether the \rightarrow_{\geq} -rule is applicable to S . We therefore have to consider the R -successors of x which are in C . Since every variable in S which is not equal to x is an R -successor of x , we compute the number $|\{z \mid z : C \in S, z \neq x\}|$. If this number is not less than the number n in $x : (\geq n R C)$, we are done since x has enough R -successors which are in C . Otherwise, constraints of the form $y_i : C$ are added to S , where y_i is a new variable.¹

Now assume that every constraint of the form $x : (\geq n R C)$ is satisfied. Thus all possible R -successors of x are already in S . Next we consider constraints of the form $x : (\leq n R C)$. If there are too many R -successors for x which are in C , we reduce them by replacing a variable by another one. Safe replacements of variables guarantee that every at-least restriction imposed on x remains satisfied. Obviously, we are confronted with another nondeterminism since we do not know which replacements may lead to a clash free complete constraint system. Thus all possibilities are tested (in the worst case) with the use of *find*.

Now assume that no more safe replacements are possible in S . Obviously, we have now reached a configuration where the \rightarrow_{\sqcap} , \rightarrow_{\sqcup} , \rightarrow_{choose} , \rightarrow_{\geq} , and $\rightarrow_{*\leq}$ -rule are not applicable to constraints containing the variable x . We now have to check whether an at-most constraint imposed on x is violated. If $x : (\leq n R C) \in S$ and $|\{z \mid z : C \in S, z \neq x\}| > n$, then there are too many R -successors of x in C , and

¹At this point one could further optimize the algorithm. Suppose $x : (\geq n_1 R C_1), \dots, x : (\geq n_l R C_l)$ are all the at-least constraints imposed on x . If there is no at-most constraint imposed on x , or if each number occurring in an at-most constraint imposed on x is not less than $n_1 + \dots + n_l$, there is no need to really introduce n_i R -successors of x which are in C_i for a constraint $x : (\geq n_i R C_i)$. In this case it is sufficient to generate exactly one R -successor as specimen.

```

satisfiable( $x, S$ ) =
  if  $\{x : \perp\} \subseteq S$  or  $\{x : A, x : \neg A\} \subseteq S$  for some primitive concept  $A$ 
    then false
  elseif  $x : C \sqcap D \in S$  and  $(x : C \notin S \text{ or } x : D \notin S)$ 
    then satisfiable( $x, S \cup \{x : C, x : D\} \setminus \{x : C \sqcap D\}$ )
  elseif  $x : C \sqcup D \in S$  and  $x : C \notin S \text{ and } x : D \notin S$ 
    then satisfiable( $x, S \cup \{x : C\} \setminus \{x : C \sqcup D\}$ ) or
         satisfiable( $x, S \cup \{x : D\} \setminus \{x : C \sqcup D\}$ )
  else let  $R = \{R \mid x : (\geq n R C) \in S \text{ for } n > 0\}$ 
    for all  $R \in R$  do:
      generate-successors( $x, R, S$ )

```



```

generate-successors( $x, R, S$ ) =
  if  $x : (\leq n R C) \in S$ ,  $|\{z \mid z \text{ is a variable in } S, z \neq x\}| > n$ ,
     $y : C \notin S$ ,  $y : \text{NNF}(\neg C) \notin S$ , and  $x \neq y$ 
    then if  $n = 0$ 
      then generate-successors( $x, R, S \cup \{y : \text{NNF}(\neg C)\}$ )
      else generate-successors( $x, R, S \cup \{y : C\}$ ) or
           generate-successors( $x, R, S \cup \{y : \text{NNF}(\neg C)\}$ )
  elseif  $x : (\geq n R C) \in S$ ,  $|\{z \mid z : C \in S, z \neq x\}| = m$  and  $m < n$ 
    then let  $y_{m+1}, \dots, y_n$  new variables:
      generate-successors( $x, R, S \cup \{y_{m+1} : C, \dots, y_n : C\}$ )
  elseif  $x : (\leq n R C) \in S$ ,  $|\{z \mid z : C \in S, z \neq x\}| > n$ 
    then let  $L$  be a list of all pairs  $(y, z)$  such that
       $y : C \in S$ ,  $z : C \in S$ ,  $y \neq x$ ,  $z \neq x$ ,  $y \neq z$ , and
      the replacement of  $y$  by  $z$  is safe in  $S$ 
      find  $(y, z)$  in  $L$  such that:
      generate-successors( $x, R, [y/z]S$ )
  elseif  $x : (\leq n R C) \in S$  and  $|\{z \mid z : C \in S, z \neq x\}| > n$ 
    then false
  else for all variables  $y \neq x$  in  $S$ :
    satisfiable( $y, \{y : C \mid y : C \in S\}$ )

```

Figure 1: A functional algorithm which decides satisfiability of concepts. The call $\text{satisfiable}(x, \{x : C\})$ returns true if and only if the concept C in negation normal form is satisfiable.

hence S contains a clash. Otherwise S is clash free w.r.t. x and our algorithm proceeds with inspecting the constraints imposed on the R -successors of x . Thus $\text{generate-successors}$ calls, for every R -successor y of x , the function satisfiable with arguments y and S' , where S' contains all the constraints imposed on y .

Theorem 4.1 *A concept C in negation normal form is satisfiable if and only if the call $\text{satisfiable}(x, \{x : C\})$ returns true.*

The complexity of satisfiable strongly depends on how the numbers occurring in at-least restrictions are coded. If we assume that these numbers are written in the decimal system as usual, the function may need space which is exponential in the size of the input concept for the following reason. Given a constraint $x : (\geq n R C)$ the function adds the constraints $y_1 : C, \dots, y_n : C$, which obviously needs exponential space in the size of the decimal representation of n . However, if we assume that the numbers are coded

unary, i.e., a number n is represented by a string of the length n , then we strongly conjecture that satisfiable needs only polynomial space in the size of the input concept.

5 REMARKS ON THE COMPLEXITY OF QUALIFYING NUMBER RESTRICTIONS

In this section we will first show that already some very small concepts of the presented language may cause long computations of the function satisfiable . Then we will discuss under which circumstances qualifying number restrictions behave better with respect to the computation time, and may thus be used in applications.

Since the satisfiability problem in our language is PSPACE-hard, one may ask whether a concept language providing qualifying number restrictions can be

used in applications. To discuss this question let us reconsider why the satisfiability (and hence the subsumption) problem in the presented language has such a high complexity.

To decide whether a concept C is satisfiable our algorithm starts with the constraint system $S = \{x : C\}$ and computes (in the worst case) all complete constraint systems issuing from S using the completion rules. One can distinguish two forms of complexities:

- Complete constraint systems issuing from S may contain exponentially many constraints. In this case, the computation of such complete constraint systems needs exponential time w.r.t. the size of C .
- There may be exponentially many different complete constraint systems issuing from S .

In [Donini et al., 1991a] it has been pointed out that both forms of complexity are present in the concept language \mathcal{ALC} . Intuitively, one can say that the first form of complexity is responsible for NP-hardness and the second form for co-NP-hardness of the subsumption problem; if both forms come together they cause PSPACE-hardness. Though the satisfiability problem in \mathcal{ALC} is PSPACE-complete, experience with an implemented system (see [Baader and Hollunder, 1990]) has revealed the following result. If an \mathcal{ALC} -concept which should be checked for satisfiability is not very long (i.e., if it contains less than 20 to 30 symbols of the form \sqcap , \sqcup , \exists , or \forall), the implemented algorithm for checking satisfiability is rather fast.²

In spite of the fact that \mathcal{ALC} and our language are similar w.r.t. worst case complexity, our language does not show the nice behaviour of \mathcal{ALC} for small concepts. It turns out that there exist concepts containing only a few symbols of the form \sqcap , \geq , or \leq , and small numbers for which the function *satisfiable* already needs a rather long time for computation. The reason for this difference seems to be that the second of the above mentioned sources of complexity has more influence for our language. This comes from the presence of the highly nondeterministic \rightarrow_{choose} - and \rightarrow_{\leq} -rules. In contrast to constraints of the form $x : C \sqcup D$, where an application of the nondeterministic \rightarrow_{\sqcup} -rule produces exactly two alternatives, constraints containing at-most restrictions in general are responsible for a lot more alternatives.

Example 5.1 Suppose that *satisfiable*($x, \{x : C\}$) is called with the concept

$$\begin{aligned} C = & (\geq 2 R (A \sqcap B)) \sqcap (\geq 2 R (A \sqcap C)) \sqcap \\ & (\leq 2 R A) \sqcap (\leq 2 R B) \sqcap (\leq 2 R C). \end{aligned}$$

²It is however not yet clear how the algorithm behaves for longer concepts which may occur in practical applications.

After applying the \rightarrow_R -rule the function *generate-successors* is called with arguments x , R , and

$$\begin{aligned} S = \{x : (\geq 2 R (A \sqcap B)), x : (\geq 2 R (A \sqcap C)), \\ x : (\leq 2 R A), x : (\leq 2 R B), x : (\leq 2 R C)\}. \end{aligned}$$

Then the constraints $y_1 : (A \sqcap B)$, $y_2 : (A \sqcap B)$, and $y_3 : (A \sqcap C)$, $y_4 : (A \sqcap C)$ are added to S to satisfy the at-least restrictions. Afterwards *generate-successors* checks whether the at-most restrictions imposed on x can be satisfied. Thus, by application of the \rightarrow_{choose} -rule for each y_i , $1 \leq i \leq 4$, $y_i : A$ or $y_i : \neg A$, $y_i : B$ or $y_i : \neg B$, and $y_i : C$ or $y_i : \neg C$ are added. Obviously, since there are 4 such successors of x , $8^4 = 4096$ alternatives are tested (in the worst case) in order to find a clash free complete constraint system.

Applications of the \rightarrow_{\leq} -rule create additional alternatives. Assume that for each i , $1 \leq i \leq 4$, $y_i : A$ is added by the \rightarrow_{choose} -rule. Since there are 4 successors of x in the concept A , *generate-successors* has to replace 2 variables to satisfy $x : (\leq 2 R A)$. In this case there are $\binom{4}{2} * \binom{3}{2} = 6 * 3$ possible replacements, which however need not all be safe.

Note that the overall number of alternatives caused by applications of the \rightarrow_{choose} - and \rightarrow_{\leq} -rule is determined by the number of R -successors of x (which depends on the at-least restrictions imposed on x), the number of at-most restrictions imposed on x , and the numbers occurring in the at-most restrictions.

This example demonstrates that—in contrast to concepts of the \mathcal{ALC} -language—already small concepts containing qualifying number restrictions cause intractable problems. This fact is not necessarily an argument in favour of excluding qualifying number restrictions from concept languages. If it turns out that in applications subsumption relations can often be computed fast with the presented algorithm, this algorithm should be employed. Nevertheless, one cannot simply rely on the fact that the input concepts are “well-behaved”.

To avoid long computations we propose the following modification of the functions *satisfiable* and *generate-successors* given in the previous section. The idea is to precompute the number of alternatives which have to be inspected in the worst case in order to find a clash free complete constraint system. The number of alternatives caused by applications of the \rightarrow_{choose} - and \rightarrow_{\leq} -rule to a variable can be estimated as demonstrated in Example 5.1. If there are too many such alternatives, then *satisfiable* stops the computation and returns *fail*. Otherwise, *satisfiable* proceeds as described in the previous section.

Thus the modified algorithm behaves as follows. If it is confronted with concepts where only few alternatives have to be inspected, then it behaves similarly to the algorithm for \mathcal{ALC} . In this case it hopefully returns the correct answer in short time. Otherwise, if there

arise too many alternatives in the computation, then it would probably take a long time to compute the correct answer. In this case the algorithm stops and returns *fail*.

Using such a modified algorithm instead of an *incomplete* algorithm—as used in almost all implemented KL-ONE systems—has the following advantage. If a fast but incomplete algorithm cannot detect a subsumption relation between two concepts, we do not know anything. A subsumption relation may or may not exist. Consequently, we do not really know whether there is *no* subsumption relation between concepts, or whether the computation of this relationship would just take too long.

On the other hand, assume that we have a subsumption algorithm as proposed above. If the algorithm returns *yes* (*no*), then we really know that there exists (does not exist) a subsumption relation. That means, both positive and negative answers are correct. Otherwise the algorithm returns the answer *fail*. However, in this case the computation of the correct answer is a hard problem, and hence fast incomplete algorithms would probably fail too.

Summing up, we have argued that one may use qualifying number restrictions although the satisfiability and subsumption problem in a concept language providing these constructs is rather hard from a computational point of view. The proposed method uses a sound and complete algorithm which in advance recognizes concepts probably causing long computations. In this case *fail* is returned. However, if the algorithm returns *yes* or *no*, then we know that this answer is *correct*.

6 CONCLUSION

The present paper is a contribution to clarifying the subsumption problem in concept languages. A sound and complete satisfiability and hence subsumption algorithm has been presented for a language providing qualifying number restrictions. We have seen that qualifying number restrictions are very expressive language constructs; they generalize role quantifications and ordinary number restrictions. Nevertheless, the presented algorithm has a very simple structure. In the following we will discuss the methodology for devising subsumption algorithms for concept languages which led to this algorithm.

The completion technique as used in the present paper was first described in [Schmidt-Schauß and Smolka, 1988] for devising a subsumption algorithm for the language *ALC*, and then extended to other concept languages (see e.g. [Hollunder, Nutt, and Schmidt-Schauß, 1990, Donini et al., 1991b]). The ideas underlying these algorithms are as follows. Since the subsumption problem can be reduced to the satisfi-

ability problem³, it is sufficient to devise algorithms which check satisfiability of concepts, i.e., which check whether a given concept denotes a nonempty set in some interpretation. If C is a concept which should be checked for satisfiability, the algorithms start with the constraint system $\{x : C\}$. Then constraints are added by applications of completion rules until either a contradiction occurs, or an interpretation I such that C^I is nonempty can be immediately obtained from the actual constraint system.

Each language construct gives rise to a particular completion rule. In general, it is relatively easy to determine for a given language construct a “corresponding” completion rule. This rule should satisfy two properties. On the one hand, it should be sound, i.e., it should satisfy one of the two properties stated in Proposition 3.3, depending on whether it is a deterministic or nondeterministic rule. On the other hand, it should be complete, i.e., if it is no longer applicable, the constraints containing the construct as uppermost operator should be “satisfied”.

Now suppose that for each language construct of the concept language we have a “corresponding” sound and complete rule. We have seen that a concept C is not satisfiable if and only if every complete constraint system (i.e., a constraint system such that no completion rule is applicable) issuing from $\{x : C\}$ contains a clash⁴. Thus, to get a decision procedure for the satisfiability problem one has to guarantee that there is no infinite chain of applications of completion rules issuing from $\{x : C\}$. In general, it is more complicated to prove termination of the completion rules than to prove soundness and completeness of these rules (see Section 3). In fact, it would be relatively easy to generate sound and complete rules for constructs which cause undecidability such as *role value maps* [Schmidt-Schauß, 1989]. However, it is clear that in this case termination cannot be guaranteed in general.

For implementation purposes one should not be content with the rule-based algorithm obtained by applying this methodology. Practical experience has shown that—for the sake of efficiency—it is important to impose an appropriate control on the rule applications. In the best case one may obtain a functional algorithm as demonstrated in Section 4.

We have pointed out that deciding satisfiability in our language has a high complexity. To avoid long computations we have modified a sound and complete satisfiability algorithm such that concepts probably causing

³Recall that C subsumes D if and only if $\neg C \sqcap D$ is not satisfiable. In [Donini et al., 1991a] it is shown how the reduction works if negation of concepts is not available in the concept language.

⁴Recall that clashes are obvious contradictions in constraint systems such as $x : A$, $x : \neg A$ or $x : \perp$.

long computations are recognized. In this case *fail* is returned. Otherwise, if the algorithm returns *yes* or *no*, then we know that—in both cases—this answer is *correct*.

Concept languages are used in KL-ONE systems to represent terminological knowledge. However, almost all of these systems have, in addition to the terminological component ("T-Box"), an assertional component ("A-Box") which allows to describe knowledge concerning particular individuals. To reason with both the T-Box and A-Box one may need algorithms for inferences such as consistency checking of the represented knowledge and "realization". It has been demonstrated in [Hollunder, 1990] that the completion technique can also be used to devise sound and complete algorithms for these inferences.

Acknowledgements

We are grateful to our colleagues Hans-Jürgen Bürkert and Werner Nutt for reading earlier drafts of the present paper. This research has been supported by the German Bundesministerium für Forschung und Technologie under grant ITW 8903 0.

References

- [Baader and Hollunder, 1990] F. Baader, B. Hollunder. *KRIS : Knowledge Representation and Inference System—System Description*. DFKI Technical Memo TM-90-03, DFKI Kaiserslautern.
- [Borgida et al., 1989] A. Borgida, R. J. Brachman, D. L. McGuinness, L. A. Resnick. "CLASSIC: A Structural Data Model for Objects." In *Proceedings of the International Conference on Management of Data*, Portland, Oregon, 1989.
- [Brachman et al., 1985] R. J. Brachman, V. Pigman Gilbert, H. J. Levesque. "An essential hybrid reasoning system: knowledge and symbol level accounts in KRYPTON." In *Proceedings of the 9th IJCAI*, pp. 532–539, Los Angeles, Cal., 1985.
- [Brachman and Schmolze, 1985] R. J. Brachman, J. G. Schmolze. "An Overview of the KL-ONE knowledge representation system." *Cognitive Science*, 9(2):171–216, 1985.
- [Dershowitz and Manna, 1979] N. Dershowitz, Z. Manna. "Proving Termination with Multiset Orderings." *C. ACM*, 22(8):465–476, 1979.
- [Donini et al., 1991a] F. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, W. Nutt. *The Complexity of Existential Quantification in Concept Languages*. DFKI Research Report RR-91-02, DFKI Kaiserslautern.
- [Donini et al., 1991b] F. Donini, M. Lenzerini, D. Nardi, W. Nutt. "Complexities in Terminological Reasoning." In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, Mass., 1991.
- [Hollunder, Nutt, and Schmidt-Schauf, 1990] B. Hollunder, W. Nutt, M. Schmidt-Schauf. "Subsumption Algorithms for Concept Description Languages." In *Proceedings of the 9th ECAI*, pp. 348–353, Stockholm, Sweden, 1990.
- [Hollunder, 1990] B. Hollunder. "Hybrid Inferences in KL-ONE-based Knowledge Representation Systems." In *Proceedings of the 14th German Workshop on Artificial Intelligence*, pp. 38–47, Eringerfeld, Germany, 1990.
- [Hollunder and Baader, 1991] B. Hollunder, F. Baader. *Qualifying Number Restrictions in Concept Languages*. DFKI Research Report RR-91-03, DFKI Kaiserslautern.
- [MacGregor and Bates, 1987] R. MacGregor, R. Bates. *The LOOM Knowledge Representation Language*. Technical Report ISI/RS-87-188, University of Southern California, Information Science Institute, Marina del Rey, Cal., 1987.
- [Nebel, 1990] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, Lecture Notes in AI, LNAI 422, Springer Verlag, 1990.
- [Nebel and Smolka, 1990] B. Nebel, G. Smolka. "Representation and Reasoning with Attributive Descriptions." In K. H. Bläsius, U. Hettstädt and C.-R. Rollinger (editors.), *Sorts and Types for Artificial Intelligence*, Lecture Notes in AI, LNAI 418, Springer-Verlag, 1990.
- [Owensnicki-Klewe, 1990] B. Owesnicki-Klewe. "A Cardinality-Based Approach to Incomplete Knowledge." In *Proceedings of the 9th ECAI*, pp. 491–496, Stockholm, Sweden, 1990.
- [Patel-Schneider, 1984] P. Patel-Schneider. "Small can be beautiful in knowledge representation." In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pp. 11–16, Denver, Colo., 1984.
- [Schmidt-Schauf, 1989] M. Schmidt-Schauf. "Subsumption in KL-ONE is undecidable." In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pp. 421–431, Toronto, Ont., 1989.
- [Schmidt-Schauf and Smolka, 1988] M. Schmidt-Schauf, G. Smolka. *Attributive Concept Descriptions with Complements*. SEKI Report SR-88-21, FB Informatik, Universität Kaiserslautern, 1988. To appear in *Artificial Intelligence*, 47, 1991.
- [Vilain, 1985] M. B. Vilain. "The restricted language architecture of a hybrid representation system." In *Proceedings of the 9th IJCAI*, pp. 547–551, Los Angeles, Cal., 1985.

A Formal Theory of Spatial Reasoning

Stephen G. Kaufman
 Department of Computer Sciences
 University of Texas at Austin
 Austin, Texas 78712
 kaufman@cs.utexas.edu

Abstract

A formalism for representing both temporal and spatial aspects of physical situations is proposed. It admits decompositions of time and space into discrete pieces, thus enabling the use of mathematical induction to reason about change in physical systems. This in turn allows properties of many systems to be inferred with a single argument. The combination of the representation and inference strategies provides a basis for reasoning about histories which is informed by the properties of the space the histories inhabit. The utility of the formalism is demonstrated by formalizing the statements that a string can pull but not push an object, and that a ratchet can turn in only one direction, and showing how these statements can be proved.

1 INTRODUCTION AND MOTIVATIONS

This paper proposes a formalism for the representation of spatial information. Additionally, the utility of mathematical induction for performing some commonsense inferences in the formalism is demonstrated. The examples include showing that a string may pull, but not push, an object; and that a ratchet wheel may turn in one direction but not the other. Both of these examples have received attention in the literature on qualitative simulation (e.g., (Forbus, Nielsen, and Faltings 1987), (Faltings 1990), (Gardin and Meltzer 1989)); in the present approach, however, conclusions are reached via deduction, not simulation.

The proposed formalization is based on an axiomatization of *tolerance spaces*; these have been studied in (Poston 1972), (Zeeman 1962), and (Dodson 1975), and were first suggested for use in AI in (Hayes 1985). A tolerance space is a collection of points with a symmetric and reflexive tolerance relation defined on it —

two points x and y so related are said to be tolerant, or close to one another, or near one another. The intuition behind the use of such spaces as models of the physical world is that at sufficiently small scales, positions or quantities are indistinguishable. According to (Poincaré 1958): "Such a statement, translated into symbols, may be written:

$$A = B, B = C, A < C.$$

"This would be the formula of the physical continuum, as crude experience gives it to us, whence arises an intolerable contradiction that has been obviated by the introduction of the mathematical continuum." However, as argued in (Hayes 1985), commonsense space is not a continuum; in particular it is not metric.

One feature of the tolerance space representation is that it allows objects and metrics (such as angle) to be decomposed into sequences. That the decomposition is a sequence allows mathematical induction to be employed in reasoning about the behavior of physical systems, if a connection between adjacent members of the sequence can be shown. The use of induction makes it possible to reason about objects with arbitrarily many components, or of arbitrary size, with one argument.

The remainder of the paper is organized as follows. The axiomatization of tolerance spaces is given in the next section, with some models for the axioms. Definitions of basic geometrical constructs are given, and some properties of them proved. Then the utility of the representation is demonstrated by formalizing both the string and the ratchet wheel, and the statements that the string can pull but not push, and that the ratchet can turn in only one direction. Sketches of proofs of these statements are also given. Finally, the work is compared to other approaches to spatial reasoning.

The following notation is used: " \supset " denotes implication, " \setminus " set difference, and " $f(A)$ ", where A is a subset of the domain of f , the image of all elements of A under f .

2 TOLERANCE SPACES

2.1 DEFINITIONS

The most essential conditions for a tolerance relation are that it be reflexive and symmetric, but *not* transitive. These conditions correspond to a point being near itself, but if a is near b and b is near c , then a is not necessarily near c (though b is near a and c is near b). Furthermore, to handle more than one dimension, each point x of the space is near several sets of points; each such set corresponds to the points near x in a particular direction (this technique was proposed in (Dodson 1975)). The following definition formalizes this:

Definition 1 An n-dimensional tolerance space is a pair (A, τ) where A is a set with a relation $\tau \subseteq A \times \mathcal{P}(A)$, such that for all $x \in A$, there exist $2n$ distinct sets related to x by τ . These are denoted by $D_i(x)$ and $\overline{D}_i(x)$ for $1 \leq i \leq n$, and satisfy the following axioms (in which $1 \leq i, j \leq n$):

Disjointness:

$$\forall x \in A [D_i(x) \cap \overline{D}_i(x) = \emptyset \wedge (i \neq j \supset (D_i(x) \cap D_j(x) = \emptyset))]$$

Symmetry:

$$\forall x, y \in A (x \in D_i(y) \supset y \in \overline{D}_i(x))$$

Containment:

$$\forall x, y \in A [y \in D_i(x) \supset D_i(x) \cup \{x\} \subset D_i(y) \cup \overline{D}_i(y) \cup \{y\}]$$

Orthogonality:

$$\forall x, y \in A [(y \in D_i(x) \wedge i \neq j) \supset \forall z \in D_j(x) \exists z' \in D_j(y) z' \in D_i(z)]$$

All the axioms hold if all occurrences of " D_i " are replaced with " \overline{D}_i " and vice-versa, or with all occurrences of " D_j " replaced with " \overline{D}_j " and vice-versa, or both. Figure 1 illustrates the two-dimensional case.

Containment ensures that there are no "holes" in the D_i . Orthogonality means that $D_i(x)$ and $D_j(x)$ really do "point" in different directions (when $i \neq j$). The various D_i are called *direction symbols*. We will use the following notation: D_i° denotes the direction symbol orthogonal to D_i , in the counterclockwise sense (so $D_1^\circ = D_2$, $D_2^\circ = \overline{D}_1$, and $\overline{D}_1^\circ = \overline{D}_2$); we will also write $D_i \perp D_i^\circ$.

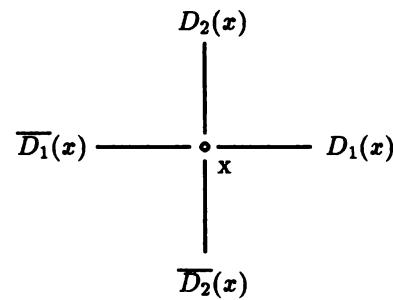


Figure 1: Tolerance relations in 2 dimensions

A class of models for these axioms are based on the first quadrant of the cartesian plane, where

$$(x_1, y_1) \in D_1((x_2, y_2)) \text{ iff } y_1 = y_2 \text{ and } 0 < x_1 - x_2 \leq f(x_2)$$

$$(x_1, y_1) \in D_2((x_2, y_2)) \text{ iff } x_1 = x_2 \text{ and } 0 < y_1 - y_2 \leq g(y_2)$$

where f and g are functions such that

$$x > y \supset f(x) + x > f(y) + y$$

$$x > y \supset g(x) + x > g(y) + y$$

Symmetry requires that $(x_1, y_1) \in \overline{D}_1((x_2, y_2))$ if $y_1 = y_2$ and $0 < x_2 - x_1 \leq f(x_1)$.

Containment holds in any such model, as follows. Let $(x_2, y_2) \in D_1((x_1, y_1))$, and fix $(x_3, y_3) \in D_1((x_1, y_1))$. We must show that

$$(x_3, y_3) \in D_1((x_2, y_2)) \cup \overline{D}_1((x_2, y_2)) \cup \{(x_2, y_2)\}$$

Equivalently, since $y_1 = y_2 = y_3$, $x_3 - x_2 \leq f(x_2)$, or $x_2 - x_3 \leq f(x_3)$, or $x_3 = x_2$. Cases:

1. $x_3 = x_2$: trivial
2. $x_3 > x_2$: $0 < x_3 - x_2 \leq f(x_1)$, or $0 < x_3 - f(x_1) + x_1 \leq f(x_1) + x_1$. Since $x_1 < x_2$, $f(x_1) + x_1 \leq f(x_2) + x_2$, so $x_3 - x_2 \leq f(x_2)$. Hence $(x_3, y_3) \in \overline{D}_1((x_2, y_2))$.
3. $x_3 < x_2$: $x_2 - x_1 \leq f(x_1)$, so $x_2 \leq f(x_1) + x_1$. Since $x_3 > x_1$, $f(x_3) + x_3 > f(x_1) + x_1$. But then $x_2 < f(x_3) + x_3$; i.e., $0 < x_2 - x_3 < f(x_3)$. Hence $(x_2, y_2) \in D_1((x_3, y_3))$.

Also, Orthogonality holds in any such model, as follows. If $(x_1, y_1) \in D_1((x_2, y_2))$ then $y_1 = y_2$. And for any $(x_2, y_3) \in D_2((x_2, y_2))$, $y_3 - y_2 \leq g(y_2)$, but then $y_3 - y_1 \leq g(y_1)$. Hence $(x_1, y_3) \in D_2((x_1, y_1))$. Since $(x_1, y_3) \in D_1((x_2, y_3))$, (x_1, y_3) is the point whose existence Orthogonality asserts. Similar arguments may

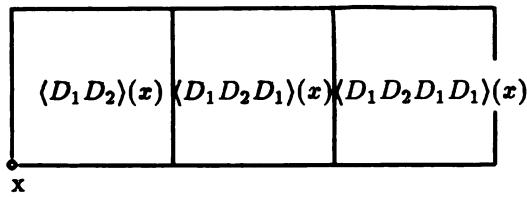


Figure 2: Directions and their direction sequences

be made for the other instances of the Orthogonality axiom.

The surface of a sphere is *not* a model (with the usual coordinates of latitude and longitude), since the points tolerant to the poles would violate orthogonality.

Notice that many tolerance relations may be defined on the same set. If τ^1 and τ^2 are two such relations on A such that $D_i^1(x) \subset D_i^2(x)$ and $\overline{D_i^1}(x) \subset \overline{D_i^2}(x)$ for all $x \in A$, then τ^1 is said to be *finer* than τ^2 , and τ^2 is said to be *coarser* than τ^1 .

We now show how various geometric primitives may be defined, and that they have appropriate properties. That we can so define these concepts makes formalizations of physical situations possible, as will be shown in section 3. In order, we will define direction, line, orthogonality, distance, and angle.

2.2 DIRECTION

Definition 2 The direction $\langle D_{i_1} \dots D_{i_k} \rangle(x)$ is the set defined recursively as:

$$k = 1 : \langle D_i \rangle(x) = D_i(x)$$

$$k > 1 : \langle D_{i_1} \dots D_{i_{k+1}} \rangle(x) = \left[\bigcup_{y \in \langle D_{i_1} \dots D_{i_k} \rangle(x)} D_{i_{k+1}}(y) \right] \setminus \langle D_{i_1} \dots D_{i_k} \rangle(x)$$

We will refer to such sets as *direction sets*, or *D-Sets*, and the sequences $\langle D_{i_1} \dots D_{i_k} \rangle$ as *direction sequences*. $\langle D_{i_1} \dots D_{i_k} \rangle$ will sometimes be abbreviated by $\langle D \rangle$. See figure 2.

We will assume that direction sequences do not contain both D_i and $\overline{D_i}$ for $1 \leq i \leq n$.

Direction sets can be empty. They have the following properties:

Lemma 1 If $w \in D_{i_0}(x)$, $y \in \langle D_{i_1} \dots D_{i_k} \rangle(w)$, and $y \notin D_{i_0}(x)$, then $y \in \langle D_{i_0} \dots D_{i_k} \rangle(x)$.

Proof: By induction on k

$k = 1$: We must show that

$$y \in \cup_{z \in D_{i_0}(x)} D_{i_1}(z) \setminus D_{i_0}(x)$$

Since we assume that $w \in D_{i_0}$, $y \in D_{i_1}(w)$, and $y \notin D_{i_0}(x)$, this is clear.

$$\begin{aligned} k > 1 : z &\in \langle D_{i_1} \dots D_{i_k} \rangle(w) && \text{(Assumption)} \\ \Rightarrow z' &\in \langle D_{i_1} \dots D_{i_{k-1}} \rangle(w) \wedge z \in D_{i_k}(z') \text{ for some } z' && \text{(Def direction)} \\ \Rightarrow z' &\in \langle D_{i_0} \dots D_{i_{k-1}} \rangle(x) && \text{(inductive hypothesis)} \\ \Rightarrow z &\in \langle D_{i_0} \dots D_{i_k} \rangle(x) \text{ where } z \in D_{i_k}(z') && \text{(Def direction)} \end{aligned}$$

Theorem 1 If $y \in \langle D_{i_1} \dots D_{i_k} \rangle(x)$, then $x \in \overline{\langle D_{i_k} \dots D_{i_1} \rangle}(y)$.

Proof: By induction on k

$$k = 1 : y \in D_{i_1}(x) \supset x \in \overline{D_{i_1}}(y) \text{ by Symmetry.}$$

$k > 1$: If $y \in \langle D_{i_1} \dots D_{i_{k+1}} \rangle(x)$, then by the definition of direction, for some $z \in \langle D_{i_1} \dots D_{i_k} \rangle(x)$, $y \in D_{i_{k+1}}(z)$ and $y \notin \langle D_{i_1} \dots D_{i_k} \rangle(x)$. By the inductive hypothesis, $x \in \overline{\langle D_{i_k} \dots D_{i_1} \rangle}(z)$. Also, $x \in \overline{D_{i_{k+1}}}(y)$ as $y \in D_{i_{k+1}}(z)$. So by lemma, $x \in \overline{\langle D_{i_{k+1}} D_{i_k} \dots D_{i_1} \rangle}(y)$, as required. \square

It turns out that direction sets are invariant under permutation of the sequence; this is a consequence of the following theorem.

Theorem 2 $\langle D_{i_1} \dots D_{i_k} D_{i_{k+1}} D_{i_{k+2}} \rangle(x) = \langle D_{i_1} \dots D_{i_k} D_{i_{k+2}} D_{i_{k+1}} \rangle(x)$

Proof: If $D_{i_{k+1}} = D_{i_{k+2}}$, trivial. So suppose that $D_{i_{k+1}} \neq D_{i_{k+2}}$.

\subset : fix

$$z \in \bigcup_{y \in \langle D_{i_1} \dots D_{i_{k+1}} \rangle(x)} D_{i_{k+2}}(y) \setminus \langle D_{i_1} \dots D_{i_{k+1}} \rangle(x)$$

(If no such z , then $\langle D_{i_1} \dots D_{i_k} D_{i_{k+2}} D_{i_{k+1}} \rangle(x)$ must also be empty, lest Orthogonality be violated). Let $z' \in \langle D_{i_1} \dots D_{i_{k+1}} \rangle(x)$ be such that $z \in D_{i_{k+2}} z'$, and $z'' \in \langle D_{i_1} \dots D_{i_k} \rangle(x)$ be such that $z' \in D_{i_{k+1}}(z'')$. By Orthogonality, since $z'' \in \overline{D_{i_{k+1}}}(z')$,

$\forall w \in D_{i_{k+2}}(z') \exists w' \in D_{i_{k+2}}(z'') : w' \in \overline{D_{i_{k+1}}}(w)$
(since $z'' \in D_{i_{k+1}}(z')$). Since $z \in D_{i_{k+2}}(z')$, $\exists w' \in D_{i_{k+2}}(z'') : w' \in \overline{D_{i_{k+1}}}(z)$. It remains to show that $w' \notin \langle D_{i_1} \dots D_{i_k} \rangle(x)$. Suppose it were: then $z \in \langle D_{i_1} \dots D_{i_{k+1}} \rangle(x)$, contradicting our assumption that $z \in \langle D_{i_1} \dots D_{i_k} D_{i_{k+1}} D_{i_{k+2}} \rangle(x)$. Hence, $z \in \langle D_{i_1} \dots D_{i_k} D_{i_{k+2}} D_{i_{k+1}} \rangle(x)$, as required.

\supset : Similar \square

The $D_i(x)$ only contain points close to x along a particular dimension. In fact the $D_i(x)$ are not tolerance relations at all, since they are not reflexive. But we can define a tolerance for 2 dimensions, as follows:

$$y \in D_{ij}(x) \text{ iff } y \in (D_i D_j)(x) \cup D_i(x) \cup \{x\}$$

There are 2^n such relations for an n -dimensional tolerance space; for $n = 2$, we have D_{ij} , D_{ij} , and D_{ij} , with the corresponding definitions. We also define the predicate Close as follows:

$$\begin{aligned} \text{Close}(x, y) \equiv & y \in D_{ij}(x) \vee y \in D_{ij}(x) \\ & \vee y \in D_{ij}(x) \vee y \in D_{ij}(x) \end{aligned}$$

2.3 LINES

Lines between two points x and y are identified with all the direction sets encountered on the direction sequence generated by the "Symmetric DDA" algorithm, used in rendering lines on a bitmapped graphic display, and described in (Newman and Sproull 1979). It is modified to produce the next element of a direction sequence, rather than plot the next point of a line.

Definition 3 *The line joining x and y is*

$$l(x, y) = \bigcup_{1 \leq p \leq k} (D_{i_1} \cdots D_{i_p})(x)$$

where $(D_{i_1} \cdots D_{i_k})$ is the direction sequence produced by the algorithm.

This notion of a line is clearly not a Euclidean one — for example, lines may intersect in more than one point. If $(D_{i_1} \cdots D_{i_k})$ is not a sequence produced by the algorithm, then $\bigcup_{1 \leq p \leq k} (D_{i_1} \cdots D_{i_p})(x)$ is called a path.

A useful property of the sequences produced by the algorithm is the following. If l is a line with n occurrences of D_1 and m occurrences of D_2 in its direction sequence, and l' is a line with $n + 1$ occurrences of D_1 and $m - 1$ occurrences of D_2 in its direction sequence, and l and l' both originate at x , then all points in l are close to a point of l' . This property will be exploited when reasoning about the ratchet.

2.4 ORTHOGONALITY

Definition 4 *The orthogonal to a given direction $(D_{i_1} \cdots D_{i_k})$ is $(D_{i_1}^o \cdots D_{i_k}^o)$ where D_i^o is orthogonal to D_i .*

We require that the "sense" of orthogonality be the same throughout. Notice that, of the set of lines joining a point p to a line l , the line orthogonal to l is among those having the fewest direction symbols. This suggests that the definition of orthogonality is the right one.

2.5 DISTANCE

The naive definition of distance would identify the distance between two points x and y as k , where $y \in (D_{i_1} \cdots D_{i_k})(x)$. But the permutation equivalence of direction sequences would then imply that the length of the hypotenuse of a triangle is equal to the sum of the lengths of its legs. The observation in the previous section shows that this naive metric has at least the property that the length of the hypotenuse of a triangle is greater than the length of either of its legs; this property will be preserved by any monotonic function of sequence length. The familiar Euclidean metric is both monotonic and Pythagorean, so we feel justified in making the following definition of distance:

Definition 5 *The distance $d(x, y)$ between points x and y is defined as:*

1. $\text{Close}(x, y) : d(x, y) < \min\{d(x, z) \mid \neg \text{Close}(x, z)\}$
2. $\neg \text{Close}(x, y) : d(x, y) = \sqrt{l_1^2 + l_2^2}$, where l_1 and l_2 are the number of occurrences of the two distinct direction symbols occurring in any sequence $(D_{i_1} \cdots D_{i_k})(y)$ such that $x \in (D_{i_1} \cdots D_{i_k})(y)$.

The first case ensures that tolerance spaces are not metric spaces, since metricity fails at sufficiently fine granularity. But for all points not close to one another, the distance metric is symmetric (by theorem 1) and satisfies the triangle inequality.

2.6 ANGLE

Definition 6 *The angle θ_k between two lines $l(x, y)$ and $l(x, z)$, where $(D_{i_1} \cdots D_{i_k})(x)$ is the direction sequence for $l(x, y)$, and $(D_{j_1} \cdots D_{j_k})(x)$ is the direction sequence for $l(x, z)$, is:*

$$\theta_k(l(x, y), l(x, z)) = \sum_{l=1}^k \frac{\Delta(D_{i_l}, D_{j_l})}{2^l}$$

where

$$\Delta(D_{i_l}, D_{j_l}) = \begin{cases} 0 & \text{if } D_{i_l} = D_{j_l} \\ 1 & \text{if } D_{i_l} \perp D_{j_l} \\ 2 & \text{if } D_{i_l} = \overline{D_{j_l}} \text{ or } \overline{D_{i_l}} = D_{j_l} \end{cases}$$

This definition of angle has the following desirable characteristics:

1. $\theta(l, l) = 0$
2. $\theta(l, \bar{l})$ maximizes $\{\theta(l, l') \mid l' \text{ is a line joined to the end of } l\}$
3. $\theta(l, l^\circ) = 1/2(\theta(l, \bar{l}))$

With this definition of angle, we may define a tolerance relation on lines as follows. Two lines $l(x, y)$ and

$l(x, z)$, whose direction sequences agree on the first k positions, are said to be *tolerant mod k* (for then $\theta_k(l(x, y), l(x, z)) = 0$). We may also define the successor of a line l (mod k) as the line whose direction sequence has a permutation differing from l 's direction sequence only at position k .

Notice that as the lines grow longer, the measure becomes more precise — just as in the commonsense world, a larger protractor is able to discern finer angles than a smaller one.

3 APPLICATIONS OF THE FORMALISM

3.1 PRINCIPLES OF FORMALIZATION

We now show how to use tolerance spaces to model the physical world. We designate a tolerance space (S, σ) to represent (empty) space, another space (T, τ) to represent time, and separate spaces for each object in the situation being modelled. For example, B may represent a block. This means that all points in B are points in the block; nothing in B gives information about the position of the block in space.

The extent of T will not be specified, but will be assumed to be long enough to encompass the actions of interest. Space and each physical object are assumed to have two dimensions, and time is assumed to have a single dimension, with a total ordering.

Positions of objects in space are given as tolerance-preserving functions from their representing spaces to S (a function from a tolerance space to another is *tolerance-preserving* if points close in the domain are close in the range). This space, S^A , is also a tolerance space; one possible tolerance on it is σ^α , defined as:

$$\langle p_1, p_2 \rangle \in \sigma^\alpha \text{ iff } \forall a \in A \text{ Close}(p_1(a), p_2(a))$$

Thus, two position functions are close if their values are pointwise near each other.

Similarly, motion is represented as a function from time to position functions; that is, a motion function is a member of $(S^A)^T$, or equivalently, $S^{A \times T}$. We may define a tolerance on this space exactly as we did for the space of position functions.

We will assume that we have a many-sorted language that includes the following sorts: Physical objects, whose representing tolerance spaces are (A, α) and (B, β) , space and time, represented by S and T ; position and motion functions, denoted by P_A and M_A for an object A ; direction symbols and sequences, denoted by D_i and $\langle D \rangle$, respectively; and natural numbers, denoted by k, n, m , etc.

Note that $P_A(A)$ and $M_A(A)(t)$ both denote regions of S . While this notation may appear to be redundant,

it is useful when the position or motion of a subset of A is of interest. Such usage will be seen many times below. The “position” of an object means the range of the position function for the object.

Using motion functions, we may define the *trajectory* of a body A over a time interval T .

Definition 7

$$\text{Trajectory}(A, T) = \bigcup_{1 \leq k \leq n, t \in T} M_A(A)(t)$$

Definition 8 An axis of symmetry of a body is a line such that for every point a in the body, there is another point b in the body such that

1. b lies in the direction orthogonal to the axis from a
2. a and b are on opposite sides of the axis
3. The distance from a to the axis is the same as the distance from b to the axis

Because of the inherent fuzziness of direction and distance, it is possible to find a sufficiently coarse tolerance relation for which an axis of symmetry exists for irregular objects. The predicate

$$\text{SymmetryAxis}(l, A, D_i)$$

asserts that l is a symmetry axis of the D-Set A , and $l = D_i(x)$ for a point in A . Note that this predicate is restricted to D-Sets.

The following predicates will also be used:

$$\text{Moving}(A, T, \langle D \rangle) \equiv$$

$$\forall t_1, t_2 \in T, a \in A (t_1 < t_2 \supset M_A(a)(t_2) \in \langle D \rangle(M_A(a)(t_1)))$$

Notice that Moving is true only for linear motion. It means that A moves in direction $\langle D \rangle$ throughout T .

$$\text{Connected}(A, B, T) \equiv \forall t \in T \exists a \in A, b \in B (\text{Close}(a, b))$$

We also define Rotating(B, T) to be equivalent to the antecedent of (1) below.

We will also use the following function constants:

$D\text{-Sets}(A, \alpha, a_0)$ denotes the D-Sets of a space A generated by a tolerance relation α with respect to a point a_0 . a_0 is usually understood to be at the “end” or “corner” of a tolerance space, like the point x in figure 1.

$DS(T)$ denotes the direction sequence $\langle D \rangle$ such that $T = \langle D \rangle(x)$ with respect to the tolerance in effect.

#occurrences($D_i, \langle D \rangle$) is the number of occurrences of the direction symbol D_i in the direction sequence $\langle D \rangle$.

$\max(\theta)$ is the maximum value the angle measure can take. It corresponds to lines with opposite directions.

Finally, we assume that appropriate tolerances are fixed for all spaces.

In the following examples, we must infer the motion of an entire body given the motion of a portion of it. For example, that a body is pushed may be formalized by stating that the pushed portion of the body is moving; the axiom below permits inferring how the whole body moves. More formally, if B is a body and C a piece of it (i.e., $C = D_{ij}(x)$ for some x on the boundary of B and appropriate i and j), then given a motion function M_C for C we must find an extension of it to a motion function M_B for B . The inferred motion may require that two bodies intersect; since this is forbidden by axiom (3) below, we may conclude by reductio that the object does not move. This means that we are obligated to determine if the motion of a body intersects any other body in the scenario.

$$\begin{aligned} [C = D_{ij}(x) \wedge x \text{ a boundary point of } B \\ \wedge \text{SymmetryAxis}(l, B, D_i) \\ \wedge l \cap C \neq \emptyset \\ \wedge \text{Moving}(C, T, \langle D \rangle) \\ \wedge 0 < \theta(\text{Trajectory}(C, T), P_B(l)) < \max(\theta) \\ \wedge \text{D-Sets}(T, \tau, t_0) = \{T_1, \dots, T_m\}] \end{aligned}$$

□

$$\begin{aligned} \exists M_B \text{ extending } M_C \\ \theta(\text{Trajectory}(C), M_B(l)(T_{k+1})) \\ > \theta(\text{Trajectory}(C), M_B(l)(T_k)) \end{aligned} \quad (1)$$

The axiom states, roughly, that pulling a rigid body tends to align the body with the direction in which it is pulled, and that pushing a rigid body tends to do the opposite. See figure 3.

The next axiom keeps physical objects in one piece:

$$\begin{aligned} A_1 \in \text{D-Sets}(A, \alpha, a_0) \wedge A_2 \in \text{D-Sets}(A, \alpha, a_0) \\ \wedge \forall a_1 \in A_1 \exists a_2 \in A_2 (a_1 \in D_{ij}(a_2)) \\ \supset \forall t \in T \forall a_1 \in M_A(A_2)(t) \\ \exists a_2 \in M_A(A_2)(t) (a_1 \in D_{ij}(a_2)) \end{aligned} \quad (2)$$

This axiom holds if D_{ij} is uniformly replaced by D_{ij} , D_{ij} , or D_{ij} . By the definition of Connected, the consequent implies that $\text{Connected}(A_1, A_2, T)$.

The next axiom states that objects may not intersect one another;

$$\forall A, B [A \neq B \supset P_A(A) \cap P_B(B) = \emptyset] \quad (3)$$

While objects may not intersect, they may certainly touch one another. "Touching" is identified with

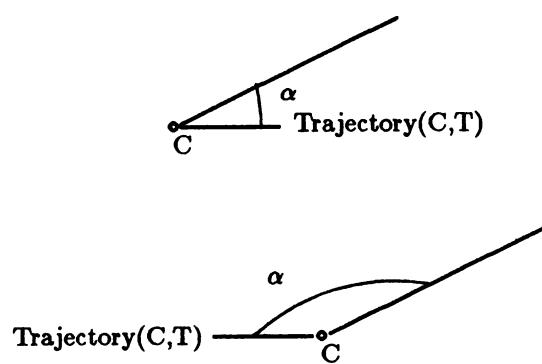


Figure 3: Illustration of Axiom 1. α increases in both cases.

object positions being close to one another, i.e., $\text{Connected}(A, B, T)$ for some T (note that "Connected" may be a permanent or temporary condition — permanent in the case of adjacent pieces of an object, temporary in the case of collisions). This explains that when objects touch, they may affect each other's motion — for any discernible change in the position of one object toward the other would require their positions to overlap. Since the axiom above forbids this, the motion of one or both must change.

3.2 Non-Monotonicity

As in any formalization of the commonsense world, conclusions must be defeasible. Consider Newton's first law of motion, that a body moving in a straight line continues to do so unless acted upon. The conclusion that an object A moves in a straight line would have to be withdrawn when an object B is added to the scenario which deflects A . It turns out that circumscription is useful for formalizing the first law of motion, as will now be shown.

First, we define the predicate

$\text{ChangeInMotion}(A, T_1, T_2)$ — A moves in a different direction throughout T_1 than throughout T_2 .

The goal is to determine the extent of Moving , minimizing the extent of ChangeInMotion .

The relevant theory contains the following axioms. First, connected things affect each other's motion:

$$\begin{aligned} [\text{Connected}(A, B, T_1) \wedge \text{Moving}(B, T_1, \langle D \rangle) \\ \wedge \text{Moving}(A, T_1, \langle D' \rangle) \wedge \langle D \rangle \neq \langle D' \rangle] \\ \supset \\ \text{Moving}(A, T_2, \langle D'' \rangle) \wedge \langle D' \rangle \neq \langle D'' \rangle \end{aligned} \quad (4)$$

$$\begin{aligned} & [\text{Moving}(A, T_1, \langle D \rangle) \wedge \text{Moving}(A, T_2, \langle D' \rangle) \\ & \quad \wedge \langle D \rangle \neq \langle D' \rangle] \\ & \supset \text{ChangeInMotion}(A, T_1, T_2) \end{aligned} \quad (5)$$

Rotation is continual change in motion:

$$\text{Rotating}(A, T_i) \supset \text{ChangeInMotion}(A, T_i, T_{i+1}) \quad (6)$$

Let A be the conjunction of (4), (5), and (6). The circumscription of ChangeInMotion in A is given by the policy:

$$\text{CIRC}[A; \text{ChangeInMotion}; \text{Moving}] \quad (7)$$

This policy implies the following:

$$\begin{aligned} \text{ChangeInMotion}(A, T_1, T_2) \equiv & \\ & [\exists B, \langle D' \rangle (\text{Connected}(A, B, T_1) \\ & \wedge \text{Moving}(A, T_1, \langle D \rangle) \\ & \wedge \text{Moving}(B, T_1, \langle D' \rangle) \\ & \wedge \langle D \rangle \neq \langle D' \rangle) \\ & \vee \text{Rotating}(A, T_1)] \end{aligned} \quad (8)$$

$$\begin{aligned} \text{Moving}(A, T_1, \langle D \rangle) \equiv & \\ & [\text{Moving}(A, T_2, \langle D \rangle) \\ & \vee \exists B, \langle D' \rangle [(\text{Connected}(A, B, T_1) \\ & \wedge \text{Moving}(B, T_1, \langle D' \rangle) \\ & \wedge \langle D \rangle \neq \langle D' \rangle)]] \end{aligned} \quad (9)$$

The last formula represents Newton's first law. As direction sequences may be empty, it also covers the case when a body is at rest. Without some positive information that something is moving, the circumscription guarantees that nothing moves; Moving and ChangeInMotion are identically false.¹

Finally, there is a problem of non-monotonicity in this framework that may not be addressed by any of the known formalisms for non-monotonic reasoning. The problem is this: The propagation of an effect over a series of entities is shown by proving that the effect is propagated from one entity to the next, and appealing to induction. Conversely, we would like to be able to infer that, if there is no sequence of entities for which the effect can be proved to propagate, then it does not propagate. This amounts to assuming that causality holds. But this inference is unjustified, since the effect may be magically transported by some other means. We will use it in the string example below, however, under the name of the *chain assumption*. The complete treatment of the string example will require a solution to this problem; the well-known relation between circumscription and mathematical induction noted in (McCarthy 1980) gives us reason to suppose that a solution exists.

¹Thanks to Vladimir Lifschitz for pointing this out.

3.3 REPRESENTING AND REASONING ABOUT THE STRING

The coordinate system of figure 1 is assumed.

Suppose that B denotes the tolerance space representing the string. That the string is long and narrow is formalized by asserting

$$\begin{aligned} \forall B_i \in \text{D-Sets}(B, \beta, b_0) \\ [\#\text{occurrences}(\text{DS}(B_i), D_2) = 1] \end{aligned}$$

We let B_1 be the D-Set closest to b_0 , and B_n be the furthest. Each member of $\text{D-Sets}(B, \beta, b_0)$ has a symmetry axis whose direction is equal to the long axis of the string, which we fix arbitrarily to be D_1 . These axes are denoted by B_{S_i} , so for each member B_i of $\text{D-Sets}(B, \beta, b_0)$ we have

$$\text{SymmetryAxis}(B_{S_i}, B_i, D_1)$$

The pulling of the string is asserted by

$$\text{Moving}(b_1, T, \langle \overline{D} \rangle)$$

where b_1 is the member of B_{S_1} that is also in $D_2(b_0)$ — i.e., the “leftmost” member of B_{S_1} . Also, $\langle D \rangle$ denotes a direction sequence such that

$$\langle D \rangle(B_1) \cap B_n \neq \emptyset$$

This means that the only case considered is when the string is pulled in the direction directly opposite its other end.

Our task is to prove that pulling one end (say B_1) of the string results in the other end of the string (B_n) moving in the same direction, and that pushing one end of the string does not result in the other end moving in the same direction. The formal statements are, for pulling,

$$\text{Moving}(B_1, T_1, \langle \overline{D} \rangle) \supset \exists T_f \text{Moving}(B, T_f, \langle \overline{D} \rangle) \quad (10)$$

and for pushing,

$$\text{Moving}(B_1, T_1, \langle D \rangle) \supset \neg \exists T_f \text{Moving}(B, T_f, \langle D \rangle) \quad (11)$$

The relationship between the motions of the symmetry axes of adjacent D-Sets of the string is at the basis of both proofs.

We must first introduce one more predicate on positions of strings, UnderTension. This object is defined on positions of objects satisfying the formula above, and states that all symmetry axes of the D-Sets of the object are aligned. This predicate is circumscribed; this attempts to capture the commonsense intuition that the relaxed state of strings is one having kinks and bends. For a string to be completely straight, it must be held. Formally:

UnderTension($P_B(B)$) \equiv
 $\forall A_{S_i}, A_{S_j} [(A_i \in \text{D-Sets}(A, \alpha, a_0) \wedge A_j \in \text{D-Sets}(A, \alpha, a_0) \wedge \text{SymmetryAxis}(A_{S_i}, A_i, D_i) \wedge \text{SymmetryAxis}(A_{S_j}, A_j, D_j) \wedge \text{Connected}(A_i, A_j)) \supset \theta(A_{S_i}, A_{S_j}) = \max(\theta)]$

Notice that UnderTension is a property of positions of strings, not strings themselves.

A proof of (10) is now sketched. By axiom (1), there exists a motion function M_B , extending M_b , such that the angle between B_{S_1} and Trajectory(b_1, T) increases. Since “increases” means a discernible increase (relative to the tolerance in effect), and there are only finitely many discernible angles, a trivial inductive argument shows that there is another motion function — the composition of the functions that are responsible for each increase in the angle — such that the angle between B_{S_1} and Trajectory(b_1) is maximal.

By the definition of Rotating, we have

$\neg\text{Rotating}(B_1, T_j)$

where T_j is the first member of $\text{D-Sets}(T, \tau, t_0)$ for which B_{S_1} is aligned. Hence if there were a point in B_1 not moving in direction $\langle \bar{D} \rangle$, (2) would be violated. That is,

$\text{Moving}(B_1, T_j, \langle \bar{D} \rangle)$

This predicate remains true for T_k after T_j , by (9), since either B_2 is rotating, or is also moving in direction $\langle \bar{D} \rangle$, as will now be shown.

The argument to this point establishes that there exists a motion function for B_1 that results in both ends of B_{S_1} moving in the same direction — i.e., the base case for the induction. The inductive case is shown as follows. Suppose that we've shown that

$\text{Moving}(B_{k-1}, T_i, \langle \bar{D} \rangle)$

The points of B_{S_k} closest to $B_{S_{k-1}}$ (the “leftmost” members of B_{S_k}) must also move in direction $\langle \bar{D} \rangle$, lest (2) be violated. Then the same argument as in the base case shows that B_{S_k} becomes aligned with $\langle \bar{D} \rangle$; it may be that $\text{Rotating}(B_{S_k}, T_i)$ holds before the alignment is complete.

By induction we conclude that

$\exists T_f \forall B_i \in \text{D-Sets}(B, \beta, b_0) \text{ Moving}(B_i, T_f, \langle \bar{D} \rangle)$

that is,

$\exists T_f \text{ Moving}(B, T_f, \langle \bar{D} \rangle)$

To prove (11), assume that

$\text{Moving}(b_1, T, \langle D \rangle)$

and observe that UnderTension is false (due to its having been circumscribed). Thus there is a pair of unaligned adjacent segments; let this pair be S_i and S_{i+1} .

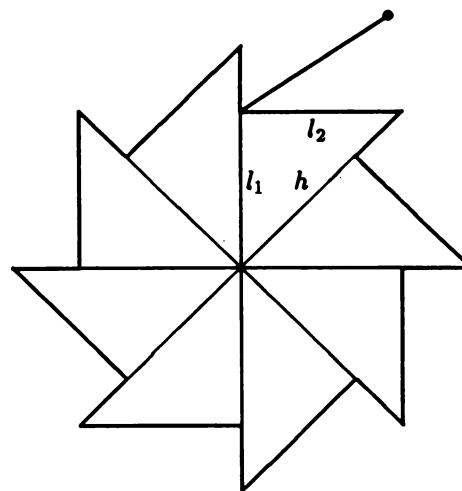


Figure 4: Ratchet and Pawl

(1) shows that the angle between them decreases, and hence by the contrapositive of lemma 2 below, the motions of opposite endpoints are not the same. Finally, the chain assumption allows us to conclude that the other end of the string does not in fact move in the same direction.

Lemma 2 *If two distinct points of a rigid body move in the same direction, then the direction of one from the other is invariant during the motion.*

There are some weaknesses in this formalization. The existence assertion in (1) sweeps a lot under the rug — for it may be the case that by the time the first symmetry axis has straightened, so has the entire string. Also, rotation of a body A connected to a body B does not cause B 's motion to change. Finally, the circumscription (7) does not consider important exceptions, such as the string being attached to a wall and not a movable object.

3.4 REPRESENTING AND REASONING ABOUT THE RATCHET

A ratchet is a device that can turn in only one direction; in figure 4, the only possible direction is counterclockwise. If the wheel turns counterclockwise, the pawl is pushed out of the way; if it turns clockwise, the pawl is eventually jammed into the corner, preventing further movement. The problem addressed here is to formalize the intuitive argument just given, so that the conclusion applies to a whole class of ratchets (with varying sizes, numbers of teeth, etc.).

As figure 4 indicates, the wheel itself may be decomposed into right triangles (recall that right triangles are formalizable in the language of tolerance spaces).

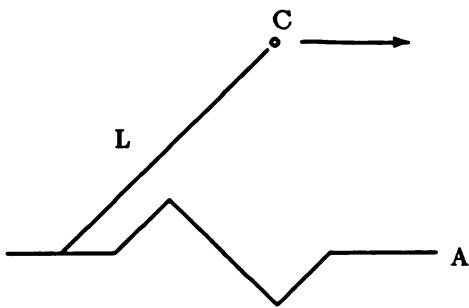


Figure 5: Pulling a pawl along a surface

The ratchet may be formalized as a sequence of position functions p_1, \dots, p_n of a right triangle with legs l_1 and l_2 and hypotenuse h . These position functions must satisfy $p_i(l_1) \subset p_{(i+1)}(\text{mod } n)(h)$

The pawl is formalized as a line of fixed length, originating at its fixed end. This captures its rigidity, length, and the fact that it has width (since lines do also).

The argument that the wheel may turn counterclockwise runs as follows. It is shown first that a pawl can be pulled along a surface, such that the pawl always remains in contact with the surface, as long as the distance between the surface and the fixed end C of the pawl is non-zero and less than the pawl's length (see figure 5). This result is sufficient to prove that the wheel can turn, provided that it can be shown that the pawl is not blocked by any other surface or object. It is sufficient because throughout the rotation of the wheel, the vertical distance between the fixed end of the pawl and the surface of the wheel is less than the length of the pawl, and the motion of the fixed end of the pawl is rightward with respect to the surface of the wheel.

That the pawl isn't blocked is established by observing that the radial surface of the ratchet is shorter than the non-radial surface, yet for any point x on the radial surface at distance d from the center, there is a point x' on the non-radial surface at the same distance from the center. But x' is further from the intersection of the radial and non-radial surfaces than is x . Hence, rotation of the wheel sufficient to raise the end of the pawl to distance d from the center of the wheel is more than sufficient to move the radial surface away from the end of the pawl, so that the pawl isn't blocked.

The formalization of pawl-pulling statement, where the coordinate system of figure 1 is assumed, is:

$$\begin{aligned}
 & [\text{Moving}(C, T, D_1) \\
 & \quad \wedge \neg \exists T_1, T_2 \text{ ChangeInPosition}(A, T_1, T_2) \\
 & \quad \wedge \exists l \in L, a \in A (\text{Close}(M_L(l)(t_0), P_A(a))) \\
 & \quad \wedge \forall t \in T \exists a \in A \\
 & \quad \quad (P_A(a) \text{ in direction } \overline{D_2} \text{ from } M_C(C)(t)) \\
 & \quad \quad \wedge 0 < d(M_C(C)(t), P_A(a)) < \text{length}(L)) \\
 & \supset \exists M_L \text{ extending } M_C \\
 & \quad (\forall a \in A \exists t \in T, l \in L \\
 & \quad \quad (\text{Close}(M_L(l)(t), P_A(a)))) \tag{12}
 \end{aligned}$$

The existence of M_L is shown by defining it inductively over the sequence T_1, \dots, T_m , as follows.

$M_L(L)(T_1)$ is simply the given initial position of L .

Let L' be $M_L(L)(T_i)$ shifted in the same direction as C is moving, by the difference between $M_C(T_{i+1})$ and $M_C(C)(T_i)$. There are three cases to consider:

1. $L' \cap A = \emptyset$ and at least one point of L' is close to A : let $M_L(L)(T_{i+1}) = L'$.
2. $L' \cap A = \emptyset$ and no points of L' are close to A : Similar to the next case.
3. $L' \cap A \neq \emptyset$: Let θ be the least angle such that the position obtained by rotating L about C by θ does not intersect A (here we use the successor relation on angles, discussed in section 2.6) Such a θ must exist, since if L is parallel to A it cannot intersect it, due to our assumption that the distance from C to A is greater than zero in the vertical direction. So we need only show that L is close to A after the rotation. Let θ' be the angle preceding θ ; the rotation of L by θ' does intersect A . By the observation in section 2.3, all points in L rotated by θ are close to a point in L rotated by θ' , and hence to a point in A .

The proof that the ratchet cannot turn clockwise is also based on an argument similar to the above, but concluding that L in figure 5 cannot be pushed to the left. In this case, (12) is modified by replacing D_1 with $\overline{D_1}$, and negating the consequent. For when the end of L is at the bottom of the “valley”, it would be stuck — the end of the pawl cannot penetrate A lest (3) be violated, and C cannot move in a circular path lest the antecedent $\text{Moving}(C, T, \overline{D_1})$ be violated.

4 Related Work

One approach to reasoning about physical systems with uncertain information is that of qualitative simulation. Well-known accounts include (Kuipers 1984), (de Kleer and Brown 1984), and (Forbus 1984). In

such systems, uncertainty in the value of real parameters is represented by *quantity spaces*, typically abstractions of the real line. Application of these techniques to spatial (multi-dimensional) reasoning was problematic until the introduction of the *place vocabulary* by Forbus (Forbus 1980), and its subsequent development by his group in (Forbus, Neilsen, and Faltings 1987), (Faltings 1987), and (Faltings 1990). A place vocabulary is a graph in which nodes denote regions of a space, and edges denote an adjacency relation between the regions. The underlying space may represent physical space (in the case of (Forbus 1980)), or in the case of (Faltings 1990), a configuration space (points of which represent configurations of a mechanism). In the latter case, one configuration may be reached from another if nodes representing them have an edge in the graph. These descriptions suffice as a basis for simulating the behavior of ratchet and escapement wheels.

The major drawback of this approach is that every new system to be simulated must have a place vocabulary computed, even if it differs only slightly (for example, an escapement wheel with one more tooth) from existing systems. This is a consequence of the requirement that the configuration description used to compute the place vocabulary be exact; it is not possible to generalize the behavior of a wheel with n teeth to one with $n+1$ teeth. Shape descriptions are thus not qualitative at all, though the simulations using them do represent scalars as qualitative magnitudes (There is also a notion of qualitative direction, corresponding to direction sequences of length 1 or 2). On the other hand, the tolerance space framework permits (indeed, requires) approximate shape descriptions; furthermore, the inferences used allow generalizing from the n -tooth ratchet to an arbitrary, similarly-constructed wheel.

Axiomatic approaches to spatial reasoning include (Shoham 1985) and (Davis 1986). The work reported herein differs in its ontology for representing time and space, and its reliance on mathematical induction as an inference method. The latter emphasis appears to be unique in the literature on spatial reasoning.

Acknowledgements

I am most grateful to Bob Boyer, Pat Hayes, and Vladimir Lifschitz for their suggestions and criticisms of drafts or of pre-text prattle. Thanks also to Boyer for providing me a wonderful environment in which to work. Most of all, I thank my wife Dara for her understanding and support during my absences.

References

Davis, E. (1986) A logical framework for solid object physics. Technical report 245, New York University Courant Institute.

de Kleer, J., and J. S. Brown (1984). A qualitative physics based on confluences. *Artificial Intelligence* 24.

Dodson, C. T. J. (1975). Tangent structure for hazy spaces. *Journal London Math. Society* 11(2):465–473.

Faltings, B. (1987). Qualitative kinematics in mechanisms. In *Proceedings of the Tenth International Conference on Artificial Intelligence*, 436–442.

Faltings, B. (1990). Qualitative kinematics in mechanisms. *Artificial Intelligence* 44(1–2): 89–119.

Forbus, K., P. Neilsen, and B. Faltings (1987). Qualitative kinematics: A framework. In *Proceedings of the Tenth International Conference on Artificial Intelligence*, 430–435.

Forbus, K. (1980). Spatial and qualitative aspects of reasoning about motion. In *Proceedings of the First National Conference on Artificial Intelligence*, 170–173.

Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence* 24: 85–168.

Gardin, F., and Meltzer, B. (1989). Analogical representations of naive physics. *Artificial Intelligence* 38(2): 139–159.

Hayes, P. (1985). "The Naive Physics Manifesto." In Hobbs, J. R., and R. C. Moore, editors, *Formal Theories of the Commonsense World*. Norwood, New Jersey: Ablex.

Kuipers, B. (1984). Commonsense reasoning about causality: Deriving behavior from structure. *Artificial Intelligence* 24: 169–203.

McCarthy, J. (1980) Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence* 13: 85–168.

Newman, W. M., and R. F. Sproull. (1979). *Principles of Interactive Computer Graphics*. 2d ed. New York: McGraw-Hill

Poincaré, H. (1958). *The Value of Science*. New York: Dover.

Poston, T. (1972). Fuzzy geometry. Ph. D. diss., University of Warwick.

Shoham, Y. (1985). Naive kinematics: One aspect of shape. In Joshi, A. (ed.), *Proceedings of the Ninth International Conference on Artificial Intelligence*, 436–442.

Zeeman, C. (1962) "The Topology of the Brain and Visual Perception." In *Topology of 3-manifolds*, edited by K. Fort. Englewood Cliffs, New Jersey: Prentice-Hall.

S4 as the Conditional Logic of Nonmonotonicity

Philippe Lamarre

I.R.I.T.

Université Paul Sabatier
118 Route de Narbonne
31062 Toulouse Cedex
email : lamarre@irit.fr
France

Abstract

We introduce a semantical approach of nonmonotonic logics in Shoham's style. It allows to underline the links between three types of apparently different formalisms: semantical approach of nonmonotonic logics, monotonic modal logics and conditional logics.

1 INTRODUCTION.

Common sense reasoning includes to draw defeasible conclusions: to add a piece of information may oblige to retract some others. Traditional logics, just because they are monotonic, don't seem able to represent this. So, nonmonotonic systems appeared to add nonmonotonic aspects to monotonic logics. Many of them have been proposed as formal models of nonmonotonicity. The best known are probably : [McDermott & Doyle 80] nonmonotonic logic I; [McCarthy 80] circumscription; [Reiter 80] default logic; [Moore 85] autoepistemic logic; and [Delgrande 88] approach based on conditional logic. An up-to-date survey may be found in [Reiter 87]. More recently, Shoham [Shoham 87a], [Shoham 87b] has proposed a general semantical approach in which many nonmonotonic logics can be represented. It is essentially his work which has inspired this paper. He presents frames (or models) constituted by the set of all interpretations of a monotonic logic structured by a preference relation. This relation is a strict partial order generally assumed to verify some property like "well founded", or "bounded".

First, we present a semantical approach of nonmonotonic logics using frames in Shoham's style, but with a little different evaluation method. This modification keeps the principal properties of Shoham's approach since under

some "well founded" hypothesis on the preference relation, they are equivalent. But as the evaluation method presented here deals directly with infinite chains, the restriction "well founded" is not necessary for this approach. To represent nonmonotonic deduction with these frames, a binary operator *preferential entailment* is used. We will see that, to study its general properties (formulas true in every preference logic captured by this semantical approach), monotonic modal logics will be good tools. Two different kind of monotonic logics present interesting arguments to be used here: modal logic with a "necessary" operator as primitive symbol (S4 will act this role), or conditional logic (a logic equivalent to S4 will be presented: C4). This induces another analogy between usual modal logics and conditional logics. Note that Boutilier [Boutilier 90], independently of the work presented here, has presented similar results on the relation between S4 and nonmonotonic logics.

2 A SEMANTICAL APPROACH FOR NONMONOTONIC LOGICS.

A monotonic logic associated with a preference relation on its interpretations is at the base of our approach. It will be used to define a *preferential entailment*, (which will be noted \Rightarrow). The basic idea is the one presented by Shoham. In a monotonic logic, to a wff A corresponds (semantically) the set of all interpretations in which A is true. In order to obtain a nonmonotonic logic, Shoham's idea is to consider not all such interpretations, but only some of them: The preferred ones (in a certain respect). So when we say A *preferentially entails* C (noted $A \Rightarrow C$) we don't consider all interpretations, but only the set of the most preferred ones where A is true. In the same way, if we consider $A \wedge B \Rightarrow C$ then we consider the set of all most preferred interpretations where A and B are true.

Considering a particular most preferred interpretation where A and B are true, there would be a preferred one where A is true and B is not. Nonmonotonicity (*A preferentially entails C* does not imply *A and B preferentially entails C*) follows from the fact that these two sets are not necessarily included. A monotonic logic and a preference relation on its interpretations define what may be called a *preference logic*. This may permit to give a semantics to some nonmonotonic formalisms which don't have any yet. But, the monotonic logic on which the semantical approach can be based needs some "good" properties (as in Shoham's approach [Shoham 87a]).

Definition 2.1 : We call *standard logic*¹ a logic \mathcal{L} with the usual compositional model-theoretic semantic (i.e. classical logic, modal logics ...). The notion of satisfiability ($\models_{\mathcal{L}}$) is defined as truth in an interpretation i (where an interpretation is a classical truth value assignment or a Kripke interpretation or ...). But, we furthermore impose the following properties to the logic: given an interpretation i ,

$$\begin{aligned} \models_{\mathcal{L}} \neg A &\text{ if and only if not } \models_{\mathcal{L}} A \\ \models_{\mathcal{L}} A \vee B &\text{ if and only if } \models_{\mathcal{L}} A \text{ or } \models_{\mathcal{L}} B \end{aligned}$$

Validity ($\models_{\mathcal{L}}$) is defined as truth in all \mathcal{L} -interpretations.

The operators *and* (\wedge) and *implication* (\rightarrow) are defined as in classical logic.

Definition 2.2 : Let \mathcal{L} be a standard logic. A *preference logic* \mathcal{L}_{\ll} based on \mathcal{L} , is then defined by the frame: $\langle I, \ll \rangle$, where I is the set of all interpretations of the logic \mathcal{L} and \ll is a preference relation on I.

Let's now present the properties we need for \ll . Intuitively, transitivity is the only one to be necessary. It corresponds to the fact that if an interpretation i_1 is preferred over i_2 and i_2 is itself preferred over i_3 then, i_1 is preferred over i_3 . We introduce reflexivity for the sake of simplicity. Given these two properties, $i_1 \ll i_2$ means i_2 is at least as preferred as i_1 . Note that, \ll is not antisymmetric, so distinct equi-preferred interpretations are allowed.

Definition 2.3 : *Satisfiability* in a preference logic \mathcal{L}_{\ll} (noted $\models_{\mathcal{L}_{\ll}}$) is defined in the following way:

If A is a well formed formula of \mathcal{L} then $\models_{\mathcal{L}_{\ll}} A$ if and only if $\models_{\mathcal{L}} A$.

Necessary (\Box) and possible (\Diamond) operators, are introduced in \mathcal{L}_{\ll} with the usual semantics:

$\models_{\mathcal{L}_{\ll}} \Box A$ iff for all interpretations i' such that $i \ll i'$, we have $\models_{\mathcal{L}} A$.

$\models_{\mathcal{L}_{\ll}} \Diamond A$ iff there is at least one interpretation i' such that

$i \ll i'$ and $\models_{\mathcal{L}} A$.

Validity is defined as satisfiability in all interpretations: A formula A is *valid* in a preference logic \mathcal{L}_{\ll} (noted $\models_{\mathcal{L}_{\ll}} A$) if and only if for any interpretation i of I we have $\models_{\mathcal{L}_{\ll}} A$.

Now, using these two operators, we are able to define what preferential entailment is. Intuitively *A preferentially entails B* if B is true in the most preferred interpretations where A is true [Shoham 87a], [Shoham 87b]. To obtain something corresponding to preferential entailment in the language, we introduce a new binary operator \Rightarrow . Let's now consider an interpretation i . In order to know if A preferentially entails B in i , we must consider all the most preferred interpretations over i where A is true. If there are none, and A is true in i , only i has to be considered. Because of reflexivity, this particular case is contained in the general one.

To explain the definition of preferential entailment, we proceed in three steps: 1 - Only linear sequences with A being true in all interpretations will be considered; 2 - We extend our study to general frames (linear or not) but always with A true everywhere; 3 - Finally we conclude considering general frames without restriction on the truth value of A. At every step, we try to find a formula corresponding to the intuitive meaning of the sentence: *B is true in the most preferred interpretations where A is true* (i.e. A preferentially entails B).

Step 1: Let's consider a linear sequence in which A is true everywhere, and let i be an interpretation of this sequence. The simplest case is when B is true for all preferred interpretations.

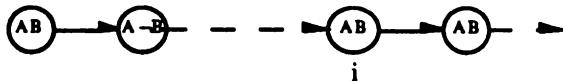


Figure 1.
(Transitivity and reflexivity are not represented.)

Then we have $\models_{\mathcal{L}_{\ll}} \Box B$. If the sequence is finite, B is true in the most preferred interpretations since it is true in all the preferred one.

Let's assume the sequence is infinite. We are not able to find explicitly the most preferred interpretation where A is true (since it is at "the end" of the infinite sequence) but, we are sure that B is true "in it", and we are not interested in anything else. So, even, in this case, we admit that *A preferentially entails B* in i .

Let's now consider a more general case : $\Box B$ is not true in i . We obtain a similar result if and only if we find an interpretation i' preferred over w satisfying $\Box B$.

¹ Here, even if, formally, the two approaches differ in some points, we use the same denomination as Shoham does.

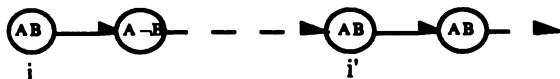


Figure 2.

In this case, we have $\vDash_{\mathcal{L}} \Diamond B$, and we admit also that A preferentially entails B in i .

Step 2: Now we consider frames looking like the following one:

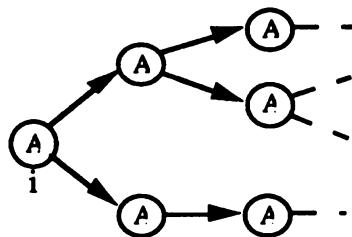


Figure 3.

For all branches (or sequences), we want to have the previous property. So, the formula needs to be modified in order to take this last remark into account. It becomes: $\vDash_{\mathcal{L}} \Box \Diamond B$.

Step 3: In general frames, there may be interpretations which don't satisfy A , but in order to check if A preferentially entails B , we are interested only in those which do. In consequence, the formula needs some last alteration such that for each preferred interpretation, if it does not satisfy A , we don't take it under consideration. So, the formula becomes: $\vDash_{\mathcal{L}} \Box (\neg A \vee \Diamond(A \wedge \Box(\neg A \vee B)))$ or equivalently $\vDash_{\mathcal{L}} \Box(A \rightarrow \Diamond(A \wedge \Box(A \rightarrow B)))$.

In the sequel, $\Box(A \rightarrow \Diamond(A \wedge \Box(A \rightarrow B)))$ will be noted $A \Rightarrow B$.

Loops, in finite sequences, just may be considered as some kind of infinite sequence.

Definition 2.4 : A preferentially entails B in a preference logic \mathcal{L}_{\leq} if and only if $\vDash_{\mathcal{L}} A \Rightarrow B$ for every interpretation i of I (i.e. $A \Rightarrow B$ is valid in the logic \mathcal{L}_{\leq}).

When there is no infinite sequence, Shoham's approach and ours, have the same intuitive foundations. In the next section, we show that they are equivalent.

Note that, in this section, writing $A \Rightarrow B$ does not induce any restriction on the form of A and B : they may be not only formulas of \mathcal{L} , but also formulas of \mathcal{L}_{\leq} (nested operators \Box , \Diamond , \Rightarrow are allowed).

Other semantics have been proposed in order to capture similar notions; but to solve the problem of infinite sequences, some more restrictions are imposed on the (preference) relation: it may be "bounded" (strict partial order) [Shoham 87a], be "well-founded" [Delgrande 87],

or verify a "smoothness condition" [Krauss & Lehmann & Magidor 90]. Since this approach directly deals with infinite sequences, the only restrictions we impose on the form of the preferential relation are those already presented i.e. transitivity and reflexivity.

3 COMPARING TO SHOHAM'S APPROACH.

In this part \Rightarrow is the only non classical operator to be under interest. Furthermore, nested operators are not allowed. So when we will write $A \Rightarrow B$, A and B will be formulas of \mathcal{L} . May be, this is more intuitive since the first aim is to have a correspondence between preferential entailment (here \Rightarrow) and nonmonotonic deduction. In this context, the use of nested operators is controversial since the understanding of something like $A \vdash (B \vdash C)$ is not clear.

Let's start with some definitions of [Shoham 87a], [Shoham 87b]:

Definition 3.1 : Let \mathcal{L} be a standard logic, and $<$ a strict partial order on \mathcal{L} -interpretations. \mathcal{L} and $<$ define a preference logic \mathcal{L}_{\leq} . An interpretation i preferentially satisfies A (written $i \vDash_{\mathcal{L}} A$) if $\vDash_{\mathcal{L}} A$ and there is no interpretation i' such that $i < i'$ and $\vDash_{\mathcal{L}} A$. In this case, i is a preferred model of A . A wff A of \mathcal{L} is preferentially satisfiable iff there is a preferred model of A . And A preferentially entails B (written $A \vdash_{\mathcal{L}} B$) if for any i , if $i \vDash_{\mathcal{L}} A$ then $i \vDash_{\mathcal{L}} B$.

Perhaps, these definitions only make sense if $<$ is "bounded" [Shoham 87a]; that is, if there is no infinite sequence $i_0 < i_1 < i_2 < \dots$. Indeed, if a wff A of \mathcal{L} is true in all these interpretations, as there is no maximal element, there is no preferred interpretation of A , and so, A is not preferentially satisfiable. This does not correspond to intuition. In consequence, it is natural to require $<$ to be "bounded". Under this restriction, we obtain the following equivalence:

Theorem 3.2 : Let \mathcal{L}_{\leq} be a preference logic defined by the preference relation $<$ ("bounded" strict partial order) on the set of all \mathcal{L} -interpretations: I . And let \mathcal{L}_{\leq} be defined like in section 2: $\mathcal{L}_{\leq} = \langle I, \ll \rangle$ with $i_1 \ll i_2$ if and only if $i_1 = i_2$ or $i_1 < i_2$. Then:

$$A \vdash_{\mathcal{L}} B \text{ iff } \vDash_{\mathcal{L}} A \Rightarrow B.$$

Proof:

First, note that \ll is defined as a transitive and reflexive relation (the former because of definition of $<$, and the latter because of $i_1 = i_2$).

Two frames have to be considered: The Shoham's one (i.e. $\langle I, < \rangle$); and ours (i.e. $\langle I, \ll \rangle$). For the sake of notations, we don't say explicitly when we speak about one or the other frame. Let's only remark that the symbol

$\vDash_{<}$ refers to Shoham's frame and that symbols \vDash_{\leq} , \Rightarrow , \Box , and \Diamond refer to ours.

The proof is in fact divided in two lemmas:

Lemma 3.3 : Let $\mathcal{L}_{<}$ be a preference logic (Shoham's one) defined by the "bounded" strict partial order $<$, and \mathcal{L}_{\leq} the corresponding frame of our approach. If $\vDash_{\leq} A \Rightarrow B$ then $A \vDash_{<} B$

Proof

Consider some i such that $i \vDash_{<} A$. By definition, we obtain (1) $\vDash_{\leq}^i A$ and (2) there is no interpretation i' such that $i < i'$ and $\vDash_{\leq}^{i'} A$. Now the hypothesis $\vDash_{\leq} A \Rightarrow B$ means that we have (for all interpretations, and in particular for i) $\vDash_{\leq} \Box(A \rightarrow \Diamond(A \wedge \Box(A \rightarrow B)))$. And then by definition of ' $<$ ' and \Box , we obtain (3) $\vDash_{\leq}^i A \rightarrow \Diamond(A \wedge \Box(A \rightarrow B))$. With (1), (3) and Modus Ponens, (4) $\vDash_{\leq}^i \Diamond(A \wedge \Box(A \rightarrow B))$ follows, and now considering (2) and the construction of ' $<$ ', it is clear that (5) $\vDash_{\leq}^i A \wedge \Box(A \rightarrow B)$, and then $\vDash_{\leq}^i B$. B is a wff of \mathcal{L} , then this is equivalent to $\vDash_{\leq}^i B$. So if $i \vDash_{<} A$ then $\vDash_{\leq}^i B$. This is true for any i such that $i \vDash_{<} A$, so $A \vDash_{<} B$.

Lemma 3.4 : Under the same hypothesis as lemma 3.3, if $A \vDash_{<} B$ then $\vDash_{\leq} A \Rightarrow B$

Proof

Consider an interpretation i such that $i \vDash_{<} A$; by \Leftarrow -construction and \Box -definition, (1) if $\vDash_{\leq}^i B$ then, since there is no $<$ -preferred interpretation i' such that $\vDash_{\leq}^{i'} A$, $\vDash_{\leq}^i A \wedge \Box(A \rightarrow B)$. Suppose now that $A \vDash_{<} B$, then we have $\vDash_{\leq}^i B$ for each interpretation i such that $i \vDash_{<} A$. The first result is: (1) if $A \vDash_{<} B$ then for any interpretation i such that $i \vDash_{<} A$, we have $\vDash_{\leq}^i A \wedge \Box(A \rightarrow B)$.

Now let's look at the hypothesis: $<$ is a "bounded" strict partial order, we can show that for any formula A and any interpretation i if $\vDash_{\leq}^i A$, then $i \vDash_{<} A$ or there is a $<$ -preferred interpretation i' such that $i' \vDash_{<} A$. Applying (1), we obtain (2): if $A \vDash_{<} B$ then for any interpretation i if $\vDash_{\leq}^i A$, then $\{ [i \vDash_{<} A \text{ and } \vDash_{\leq}^i A \wedge \Box(A \rightarrow B)] \text{ or there is a } <\text{-preferred interpretation } i' \text{ such that } [i' \vDash_{<} A \text{ and } \vDash_{\leq}^i A \wedge \Box(A \rightarrow B)] \}$. The "then" part simplifies in $\vDash_{\leq}^i A \wedge \Box(A \rightarrow B)$ or there is a $<$ -preferred interpretation i' such that $\vDash_{\leq}^i A \wedge \Box(A \rightarrow B)$. If we speak about \Leftarrow -preferred interpretations instead of $<$ -preferred, this becomes: $\vDash_{\leq} \Diamond(A \wedge \Box(A \rightarrow B))$, and then, considering the whole expression (2), we obtain: if $A \vDash_{<} B$ then for any interpretation i , $\vDash_{\leq}^i A \rightarrow \Diamond(A \wedge \Box(A \rightarrow B))$. Since it is true for all interpretations, we can equivalently write: for any interpretation i , $\vDash_{\leq}^i \Box(A \rightarrow \Diamond(A \wedge \Box(A \rightarrow B)))$;

looking at \Rightarrow -definition. We finally obtain: if $A \vDash_{<} B$ then $\vDash_{\leq} A \Rightarrow B$.

Lemmas 3.3 and 3.4 give the proof of theorem 3.2.

The consequence is that all the nonmonotonic logics that can be characterized in terms of Shoham's preference logics can also be characterized by the section 2 approach. Furthermore, two very useful properties are now allowed: first, we can use equi-preferred interpretations (two or more). This case has also been studied by [Brown & Shoham 88] with *biazed logics*. And, secondly, we do not have to worry about infinite chains.

This semantical approach enables to characterise some nonmonotonic logics and to give them some semantical aspects. Now, it would be interesting to study common properties of nonmonotonic logics which we are able to deal with. As the frames we have defined, and Kripke models for normal modal logics have very similar forms, it seems interesting to use such a logic for this study. But another approach is possible. We are interested in a nonmonotonic binary relation (preferential entailment), and conditional logics are specialised in this kind of operators. These two approaches are interesting: the former because we use a necessary notion in part 2 to present and to define preferential entailment, and automated theorem proving methods are known for normal modal logics. The latter is also interesting because it is a more direct approach of preferential entailment properties.

4 RELATING THIS APPROACH WITH MODAL LOGIC S4.

In the whole following, we will be concerned only with nonmonotonic logics defined on propositional calculus.

Among the modal logics on the start line, the first candidate is S4: its semantics is also based on transitive and reflexive relations.

We recall briefly the axiomatic and semantics of S4:

Definition 4.1 : Logic S4

Axiomatic consists the following axiom schemata:

- A1 - $(A \vee A) \rightarrow A$
- A2 - $B \rightarrow (A \rightarrow B)$
- A3 - $(A \vee B) \rightarrow (B \vee A)$
- A4 - $(A \rightarrow C) \rightarrow ((A \vee B) \rightarrow (A \vee C))$
- A5 - $\Box A \rightarrow A$
- A6 - $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$
- A7 - $\Box A \rightarrow \Box \Box A$

and inference rules:

- MP - $\frac{A \quad A \rightarrow B}{B}$
- N - $\frac{A}{\Box A}$

Semantics:

An S4-model is a triple $M = \langle W, R, v \rangle$ where:

- _ W is a non empty set of worlds, and
- _ R is a reflexive and transitive relation on W , and
- _ v is a valuation from the set of worlds and propositional variables to {true, false}.

$\models_M p$ iff $v(w,p) = \text{true}$ (p is a Propositional Variable)

$\models_M A \wedge B$ iff $\models_M A$ and $\models_M B$

$\models_M \neg A$ iff not $\models_M A$

$\models_M \Box A$ iff for any w' such that $w R w'$ we have $\models_M A$

The valuation v may be seen as a function from the set of worlds to the set of classical interpretations. For more information on S4, see [Hughes & Cresswell 68]. The notions of satisfiability (\models_M), validity (\models_{S4}) and theoremhood (\vdash_{S4}) are defined as usual. We define \Diamond and \Rightarrow respectively by: $\neg\Box\neg A$, and $\Box(A \rightarrow \Diamond(A \wedge \Box(A \rightarrow B)))$. " $\Diamond A$ " is read *A is possible* and " $A \Rightarrow B$ " may be read *Normally if A then B or It is usually admitted that if A then B*, but for the sake of coherence, here, we will read it: *A preferentially entails B*. Note that, as expected, \Rightarrow is a non monotonic operator ($A \Rightarrow C$ does not entail $A \wedge B \Rightarrow C$) even if it is defined using monotonic operators in a monotonic logic.

Remember that our aim is to link the approach developed in part 2 with S4. Given a frame $\langle I, \ll \rangle$ of a preference logic L_{\ll} , it is not difficult to find an intuitive transformation to an S4-model since the only difference is the interpretation function.

Definition 4.2 : So, if $\langle I, \ll \rangle$ is a frame of a preferential logic, we will define the corresponding S4-model $M = \langle W_s, R_s, v_s \rangle$ where $W_s = I$, $R_s = \ll$, $v_s(w,p) = \text{true}$ iff $\models_{PC} p$ (p is a propositional variable, PC is Propositional Calculus).

Theorem 4.3 : Given a well formed formula (with or without modal operators) A , $\models_{\ll} A$ if and only if M defined as above, for all worlds w in W_s , $\models_M A$.

(and so $\models_{\ll} A \Rightarrow B$ if and only if for all worlds w in W_s , $\models_M A \Rightarrow B$).

Proof is trivial.

And we obtain :

Corollary 4.4 : If A and B are wff (with or without modal operators) then if $\models_{S4} A \Rightarrow B$ (or $\vdash_{S4} A \Rightarrow B$) then in any preference logic L_{\ll} , we have $\models_{\ll} A \Rightarrow B$.

This has been obtained because, considering a frame of part 2, we are able to construct an S4-model such that the frame and the S4-model satisfy the same formulas. But in order to obtain the reverse, i.e. if in any preference logic L_{\ll} , we have $\models_{\ll} A \Rightarrow B$ then $\models_{S4} A \Rightarrow B$, the definition of a frame given in part 2 needs to be extended. We no longer consider the set of all interpretations, but (as in S4) a non empty set of worlds W . And to link worlds and interpretations, we could use a function noted I from W

to I . Then the new frame has the following form: $\langle W, \ll, I \rangle$. This modification seems to be reasonable and natural. In fact, this kind of frames is already known. It have been used by Kraus, Lehmann and Magidor to prove their important representation theorem for Preferential Consequence Relations [Krauss & Lehmann & Magidor 90]. Boutilier [Boutilier 90] has also proved, in a parallel work to this paper, a correspondence between S4 and Kraus, Lehmann and Magidor's frames using an equivalent definition of conditional ($A \Rightarrow B =$ Boutilier's Def $\Box(\Box\neg A \vee \Diamond(A \wedge \Box(\neg A \vee B)))$).

So S4 expresses common properties of preference logics. To refer to Bell's work [Bell 90], which gives another correspondence, a little less strong than here, between a conditional logic named C and Shoham's approach, we assert that S4 is *the logic of nonmonotonicity*.

In the next part, we give a direct approach of preferential entailment using a conditional logic.

5 RELATING THIS APPROACH WITH A CONDITIONAL LOGIC.

Generally, conditional logics are used, to formalise and study properties such as relevance and necessity ([Dunn 86], [Anderson & Nuel & Belnap 75]). More recently, they have been applied to nonmonotonicity. The main idea is to introduce a nonmonotonic binary operator in a monotonic logic in order to obtain a monotonic reasoning about nonmonotonic relation. Different results may be found in [Nute ??], [Nute 84], [Stalnaker 68], [Delgrande 87]. Here, we are going to define a conditional logic expressing general properties of preferential entailment in a way similar to [Bell 90].

5.1 THE LOGIC C4.

Definition 5.1 : C4 axiomatic

It consists of the following axiom schemes:

AC1, AC4 - PC axioms

AC5 - $A \Rightarrow A$

AC6 - $((A \Rightarrow B) \wedge (A \Rightarrow C)) \rightarrow (A \Rightarrow (B \wedge C))$

AC7 - $((A \Rightarrow C) \wedge (B \Rightarrow C)) \rightarrow ((A \vee B) \Rightarrow C)$

AC8 - $((A \Rightarrow B) \wedge ((A \wedge B) \Rightarrow C)) \rightarrow (A \Rightarrow C)$

AC9 - $((A \Rightarrow B) \wedge (A \Rightarrow C)) \rightarrow ((A \wedge B) \Rightarrow C)$

AC10 - $(\neg A \Rightarrow A) \rightarrow A$

AC11 - $(A \Rightarrow B) \rightarrow (C \Rightarrow (A \Rightarrow B))$

AC12 - $A \Rightarrow ((A \Rightarrow B) \rightarrow (A \Rightarrow B))$

AC13 - $A \Rightarrow ((A \Rightarrow C) \rightarrow (A \wedge B \Rightarrow C))$

AC14 - $A \Rightarrow (\neg(A \Rightarrow B) \rightarrow (C \Rightarrow (A \rightarrow (A \Rightarrow B))))$

and inference rules

MP - $\frac{A \quad A \Rightarrow B}{B}$

RCM - $\frac{B \rightarrow C}{(A \Rightarrow B) \rightarrow (A \Rightarrow C)}$

Axioms AC11-14 deal with nested formulas. This may look contradictory as, in part 3, we argue to neglect formulas with nested operators. Moreover, it is clear that the interesting property of nonmonotonicity of

preferential entailment is lost when nested operators are concerned. In fact, we only need these axioms to obtain full correspondence with S4.

Note that this axiomatic is in the style of, for example, logic NP [Delgrande 87&88]. But C4 does not subsume NP, neither it is subsumed by it.

C4, is normal in term of [Chellas 75], that is, it is closed under the rules:

$$\text{RCEA} : \frac{A \leftrightarrow B}{(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)}$$

$$\text{RCK} : \frac{(B_1 \wedge \dots \wedge B_n) \rightarrow B}{((A \Rightarrow B_1) \wedge \dots \wedge (A \Rightarrow B_n)) \rightarrow (A \Rightarrow B)}$$

Proof is in annex.

From these two rules, it follows that equivalent propositions can be substituted into antecedent or consequent of a conditional variable. And by induction, we obtain full substitution of equivalents.

In the semantics, we use a selection function as done often in conditional logics see, for example, [Stalnaker 68] or [Chellas 75] or [Delgrande 87&88].

Definition 5.2 : C4-semantics: A C4-model is a triple $M = \langle W, f, v \rangle$ where W is a non empty set of worlds, f a selection function from $W \times 2^W$ to 2^W and v a valuation from $W \times P$ to {true ; false}. Intuitively W is the set of possible worlds; given a world w and a set of worlds S the selection function f associates the subset of S of the most preferred worlds over w . For a wff A , $|A|_M$ stands for the subset of worlds in M in which A is true. If there is no ambiguity on M , we will adopt $|A|$. $\models_M A$ will be used to assert that A is true in model M at world w . Satisfiability is defined by:

$\models_M p$ iff $v(w, p) = \text{true}$.

$\models_M \neg A$ iff $\neg \models_M A$

$\models_M A \wedge B$ iff $\models_M A$ and $\models_M B$

$\models_M A \Rightarrow B$ iff $f(w, |A|) \subseteq |B|$

where the function f has the following properties:

f5 - $f(w, |A|) \subseteq |A|$

f7 - $f(w, |A \vee B|) \subseteq (f(w, |A|) \cup f(w, |B|))$

f8 - If $f(w, |A|) \subseteq |B|$ then $f(w, |A|) \subseteq f(w, |A \wedge B|)$

f9 - If $f(w, |A|) \subseteq |B|$ then $f(w, |A \wedge B|) \subseteq f(w, |A|)$

f10 - If $f(w, |A|) = \emptyset$ then $\models_M A$

f11 - If $w_1 \in f(w, |C|)$ then $f(w_1, |A|) \subseteq f(w, |A|)$

f12 - If $w_1 \in f(w, |A|)$ then $w_1 \in f(w_1, |A|)$

f13 - If $w_1 \in f(w, |A|)$ then $f(w_1, |A \wedge B|) \subseteq f(w_1, |A|)$

f14 - If $w_1 \in f(w, |A|)$ and $w_2 \in f(w_1, |C|) \cap |A|$ then

$f(w_1, |A|) \subseteq f(w_2, |A|)$

The numeration of these properties corresponds to the axiomatic's one. Note that ACn^i corresponds to fn^i .

Although f8 and f9 reduce in f8': "If $f(w, |A|) \subseteq |B|$ then $f(w, |A|) = f(w, |A \wedge B|)$ ", we do not adopt this abbreviation to keep the correspondence between

semantics and axiomatic, and to conserve some granularity on properties.

Validity is defined by: $\vdash_{C4} A$ iff for any C4-model M and any world w of it, $\models_M A$.

Theorem 5.3 : C4-semantics is sound and complete in respect with C4-axiomatic, i.e. $\vdash_{C4} A$ iff $\vdash_{C4} A$.

Proof

Two parts must be considered. First: C4-theorems are true in every C4-models, i.e. all axioms are true in every models and inference rules preserve validity. This is not difficult to verify since there is a corresponding rule for every axiom. Secondly, we need to prove that if a formula is true in all C4-models, then it is a C4-theorem. This last part is proved, as in [Chellas 75] for some other conditional logics, using the canonical model method. The canonical model of C4, noted CM, is defined as follows:

- _ $W = \{ w \mid w \text{ is a maximal consistent set of wffs}^2 \}$
- _ $f(w, |A|)_M = \{ w_1 \in W \mid \{ B \mid A \Rightarrow B \in w \} \subseteq w_1 \}$
- _ $v(w, p) = \text{true iff } p \in w$.

Where $|A|_M = \{ w \mid w \in W \text{ and } A \in w \}$.

In a first step we prove that for any wff A and any $w \in W$: $\models_M A$ iff $A \in w$. This is made by induction on the form of the formula A .

In a second step, we need to verify that CM is a C4-model i.e. that every condition on f is verified. The details of the whole proof can be found in [Lamarre 90].

We have now at our disposal conditional logic C4 whose semantics given by definition 5.2 is sound and complete with respect to its axiomatic given by definition 5.1.

5.2 RELATIONS BETWEEN C4 AND S4; C4 AND PREFERENTIAL ENTAILMENT

Let's now study the correspondence of C4 conditional operator and preferential entailment. We proceed by showing that C4 and S4 constitute in fact the same logic, so that we can use the results of part 4.

A first intuitive obstacle may arise when giving to C4 the same expressive power as S4 defining *necessary* (\Box) and *possible* (\Diamond) as follows.

Definition 5.4 : $\Box A =_{\text{Def}} \neg A \Rightarrow A$ and $\Diamond A =_{\text{Def}} \neg(\neg A \Rightarrow A)$.

Intuitively, $\neg A \Rightarrow A$ means that A is true in the

² Wffs are formed here on the Propositional Calculus language augmented with the binary operator \Rightarrow . Nested formulas are allowed.

"preferred" worlds³ where $\neg A$ is; i.e. $f(w, \neg A) \subseteq IAI$. But by property f5, $f(w, \neg A) \subseteq \neg AI$, and then $f(w, \neg A) \subseteq IAI \cap \neg AI = \emptyset$, so $f(w, \neg A) = \emptyset$; and then by property f10, $\models_{\mathbb{M}} \neg A$. Using property f11, this may be repeated for all w_1 such that $w_1 \in f(w, IC)$. So, we can say that as far as w is concerned, for all "preferred" worlds, A is true. The \Diamond -definition can be explained in the same intuitive way. Considering these two new definitions, C4-language may be defined as the Propositional Calculus one augmented with the binary operator \Rightarrow , and the two unary operators: \Box and \Diamond . Note that, now, S4 and C4 have the same language.

But the similarity is greater: they have the same set of theorems.

Theorem 5.5 : $\vdash_{C4} A$ if and only if $\vdash_{S4} A$.

To prove this theorem, we are going to restrict the languages of the two logics in order to obtain a common language based on only one modal operator. So that in the following proofs, when some inductive reasoning on the construction of a formulas will be used, we will only have one non classical operator to consider. And then, the proofs will be more straightforward. Two choices are possible: to keep S4-necessary or C4-entailment. Our preference goes to C4-entailment for two main reasons. First it is what we are studying here, and it is a good way to understand better its behaviour. The second argument is a technical one: it would be difficult to work with necessary in C4 because of its definition. We don't claim here that the definition of entailment in S4 is very simple, but maybe we are more accustomed to manipulate Kripke models than conditional ones.

Trivially, all C4 formulas may be expressed using only entailment as modal operators since it is a primitive operator.

It is also possible to do the same in S4. We only have to note that $\vdash_{S4} \Box A \leftrightarrow \Box(\neg A \rightarrow \Diamond(\neg A \wedge \Box(\neg A \rightarrow A)))$ or equivalently: $\vdash_{S4} \Box A \leftrightarrow (\neg A \Rightarrow A)$. (The proof may be found in the annex.) From this and full substitution of equivalents, we are able to replace every formula containing some necessary operator by an equivalent expression using entailment. So each well formed formula of S4 has an equivalent form which uses only one single operator: entailment.

Then S4 language may be restricted to Propositional calculus one augmented by the binary operator \Rightarrow .

Lemma 5.6 : If $\vdash_{S4} A$ then $\vdash_{C4} A$

Proof

³ Here, we say that a world w_1 is "preferred" over w if $w_1 \in f(w, IC)$ for some wff C

By theorem 5.3, and the corresponding theorem for S4, this is equivalent to: If $\models_{S4} A$ then $\models_{C4} A$.

I.e.: if for any S4-model M_s and any world w of this model $\models_{M_s} A$ then for any C4-model M_c and any world w of this model, $\models_{M_c} A$.

If we take the contrapositive, it becomes: If there is a C4-model M_c and a world w such that not $\models_{M_c} A$, then there is an S4-model M_s and a world w such that not $\models_{M_s} A$.

Since in S4 as in C4, not $\models_{M_c} A$ is the same as $\models_{M_c} \neg A$, this can be rewritten:

If there is a C4-model M_c and a world w such that $\models_{M_c} \neg A$, then there is an S4-model M_s and a world w such that $\models_{M_s} \neg A$.

If we can prove this last sentence for any A , then, we would have a proof of lemma 5.6.

Given a C4-model $M_c = \langle W_c, f_c, v_c \rangle$, we construct the following model: $M_s = \langle W_s, R_s, v_s \rangle$ with:

– $W_s = W_c$ (they will be noted simply W).

– $v_s = v_c$ (They will be noted simply v).

– R_s (noted R in the following) is defined in the following way:

for any w_1, w_2 of W : $w_1 R w_2$ iff $w_1 = w_2$ or $w_2 \in f(w_1, IX)$ for some X .

The construction of R seems to be natural. M_s is an S4 model. To prove it we have to verify that R as defined above is reflexive and transitive. See [Lamarre 90].

Furthermore, we prove: For any w in W $\models_{M_c} A$ iff $\models_{M_s} A$. Trivially, this is true for every formula which does not have any conditional operator. For formulas which contain such an operator, the proof is by induction on the number of nested conditional operators. Two parts are necessary: 1 - if $\models_{M_c} A \Rightarrow B$ then $\models_{M_s} A \Rightarrow B$; and 2 - if $\models_{M_c} \neg(A \Rightarrow B)$ then $\models_{M_s} \neg(A \Rightarrow B)$ or equivalently if not $\models_{M_c} A \Rightarrow B$ then not $\models_{M_s} A \Rightarrow B$, or in another form if $\models_{M_c} A \Rightarrow B$ then $\models_{M_s} A \Rightarrow B$. These two parts give $\models_{M_c} A \Rightarrow B$ iff $\models_{M_s} A \Rightarrow B$. This is very fastidious; the whole proof is not detailed here, but may be found in [Lamarre 90].

Lemma 5.7 : If $\vdash_{C4} A$ then $\vdash_{S4} A$

Proof

As for the previous lemma, we will use semantics and construct a corresponding C4-model M_c for any S4-model M_s , such that M_c and M_s will satisfy the same formulas. S4 verifies the finite model property [Hughes & Cresswell 84] and this is a very interesting property. Indeed, for every S4 model M , by a filtration method, we are able to produce another model M^* which satisfies the same formulas, but in which W^* is finite. S4 is then characterised by models having a finite set of worlds W .

So, here only finite models have to be considered.

Let $M_s = \langle W_s, R_s, v_s \rangle$ be a finite S4-model. We construct model $M_C = \langle W_c, f_c, v_c \rangle$ such that:

- _ $W_c = W_s$ (noted simply W).
- _ $v_c = v_s$ (noted simply v).
- _ f_c (noted simply f) is defined in the following way:

Given a world of W and a well formed formula:

$f(w, |A|) = \{ w' \mid wRw' \text{ and } \models_{Ms} A \text{ and for any } w'' \text{ if } w'Rw'' \text{ and } \models_{Ms} A \text{ then } w''Rw' \}$

We first prove that M_c is a C4-Model by verifying all properties of f . Then, we show that that M_s and M_c satisfy the same formulas. The whole proof may be found in [Lamarre 90].

Proof of theorem 5.5: By lemmas 5.6 and 5.7.

Considering theorem 5.5, C4 and S4 constitute in fact the same logic. In this sense we may affirm that S4 is a conditional logic. Boutilier ([Boutilier 90]) has already noted this, but, providing an axiomatic (and semantics) equivalent to S4, in a pure conditional style, is a step more in this direction.

6 SOME PROPERTIES OF PREFERENTIAL ENTAILMENT.

Now, our purpose is to use S4 and C4 to study properties of preference logics.

First let's have a look at figure 4. It expresses the links between, on the one hand preferential entailment, and on the other hand material (or classical) implication (\rightarrow), and also Lewis's strict implication (\supset).

$$A \supset B \underset{\text{Def}}{=} (A \rightarrow B)$$

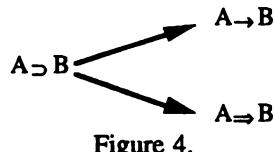


Figure 4.

Arrows "... → ..." stand for "if...then...".

Remember that material implication means that, for a given world, if A is true so is B . So in the point of view of Lewis's strict implication, which speaks about all accessible (preferred) worlds, the two others entailments formalise opposite notions. Material implication speaks about the nearest world: itself, and preferential entailment about the most distant worlds (with respect to the truth of A).

Theorem 6.1 : Some results on \Rightarrow .

$A \Rightarrow A$	Reflexivity
$\frac{A \Leftarrow B \quad A \Rightarrow C}{B \Rightarrow C}$	Left Logical Equivalence
$\frac{A \rightarrow B \quad C \Rightarrow A}{C \Rightarrow B}$	Right Weakening
$\frac{A \wedge B \Rightarrow C \quad A \Rightarrow B}{A \Rightarrow C}$	Cut
$\frac{A \Rightarrow B \quad A \Rightarrow C}{A \wedge B \Rightarrow C}$	Cautious Monotonicity
$\frac{A \Rightarrow B \quad B \Rightarrow A \quad A \Rightarrow C}{B \Rightarrow C}$	Equivalence
$\frac{A \Rightarrow B \quad A \Rightarrow C}{A \Rightarrow C}$	And
$\frac{A \Rightarrow B \wedge C}{A \vee B \Rightarrow C}$	Or
$\frac{A \wedge B \Rightarrow C}{A \Rightarrow B \rightarrow C}$	S
$\frac{A \wedge \neg B \Rightarrow C \quad A \wedge B \Rightarrow C}{A \Rightarrow C}$	D

Proofs, which use logic C4, are in the annex. We may note that the proofs are not difficult. The same proofs done with S4 are longer and more difficult. Note also that only the flat part of S4 (no nested operator) is used.

Due to the inference rules of theorem 6.1, \Rightarrow may be considered, in terms of [Krauss & Lehmann & Magidor 90], as a *preferential consequence relation*. This holds in C4, in S4 and by corollary 4.4 in every logic which may be characterised by some frame of part 2. This result corresponds to Boutilier's one when he proves, with an equivalent definition of preferential entailment, the correspondence between S4 "flat" conditional part and the Kraus, Lehmann and Magidor's approach.

7 CONCLUSION

Even if we are not able to characterise all nonmonotonic logics, S4 and C4 can be used to study the common properties of nonmonotonic preferential deductions. Indeed, if we consider the default logic of Reiter [Reiter 80], with nonmonotonic deduction defined as truth in some extension, it trivially does not satisfy the rule And of theorem 6.1. It follows that it is not a preferential logic [Makinson 88], and it can't be characterised in terms of part 2.

Other normal modal logics may be useful to study some other nonmonotonic operators. For example the logic S4.3, which is an extension of S4, brings to our definition a new interesting property that is rational monotonicity: $\frac{\neg(A \Rightarrow B) \quad A \Rightarrow C}{A \wedge \neg B \Rightarrow C}$ (See [Boutilier 90] or [Lamarre 90]). This property has been studied for example in [Lehmann 89], and the corresponding axiom is an axiom of the logic NP [Delgrande 87 & 88]. As suggested by Boutilier, other definitions of the conditional operator may also be studied to obtain some different properties.

The link between S4 (or more generally monotonic logics) and nonmonotonicity is principally established by the semantics. But it is not a total correspondence. When we speak about a nonmonotonic logic, as in part 2, we use one and only one frame (or model). Only formulas which are true in this particular model are under our interest. But when we speak about logic S4, (or C4), all models are considered (validity is defined as truth in all models), and we have as theorems the formulas which are true in all models. So, using S4, we capture only the formulas which are true in each preference logic. If we want to characterise theorems of a particular logic, S4, or C4, are not sufficient.

However, we may extend the results of this paper by searching for a set of formulas Γ such that $A \vdash B$ in the considered preferential logic if and only if $\Gamma \models_{S4} A \Rightarrow B$, or $\models_{S4} \Gamma \rightarrow (A \Rightarrow B)$. Then, the set Γ would become the characterization of a nonmonotonic preferential logic.

ANNEX: proofs

C4 is closed under the rules RCEA and RCK.

1°) RCEA

- AC9,PC (1) $(A \Rightarrow B) \rightarrow ((A \Rightarrow C) \rightarrow ((A \wedge B) \Rightarrow C))$
- AC8,PC (2) $(A \Rightarrow B) \rightarrow (((A \wedge B) \Rightarrow C) \rightarrow (A \Rightarrow C))$
- (1),(2),PC (3) $(A \Rightarrow B) \rightarrow (((A \wedge B) \Rightarrow C) \leftrightarrow (A \Rightarrow C))$
- (3)[AIB; BIA] (4) $(B \Rightarrow A) \rightarrow (((B \wedge A) \Rightarrow C) \leftrightarrow (B \Rightarrow C))$
- (3),(4),PC (5) $((A \Rightarrow B) \wedge (B \Rightarrow A)) \rightarrow ((A \Rightarrow C) \leftrightarrow (B \Rightarrow C))$

Let's assume that $A \leftrightarrow B$

Then $A \rightarrow B$, and by RCM $(A \Rightarrow A) \rightarrow (A \Rightarrow B)$, which give using AC5 and MP: $(A \Rightarrow B)$. Furthermore, $B \rightarrow A$. So by the same way $(B \Rightarrow A)$.

These two last results and (5) with MP give: $((A \Rightarrow C) \leftrightarrow (B \Rightarrow C))$.

So if $A \leftrightarrow B$, then $(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)$.

2°) RCK

If $n=1$ then RCK is RCM.

Assume that RCK is true for some given n .

So $\frac{(B_1 \wedge \dots \wedge B_n \wedge B_{n+1}) \rightarrow B}{((A \Rightarrow B_1) \wedge \dots \wedge (A \Rightarrow (B_n \wedge B_{n+1})) \rightarrow (A \Rightarrow B))}$.

Furthermore AC6 gives $((A \Rightarrow B_n) \wedge (A \Rightarrow B_{n+1})) \rightarrow (A \Rightarrow (B_n \wedge B_{n+1}))$.

So if $((A \Rightarrow B_1) \wedge \dots \wedge (A \Rightarrow (B_n \wedge B_{n+1})) \rightarrow (A \Rightarrow B))$ then $((A \Rightarrow B_1) \wedge \dots \wedge (A \Rightarrow B_n) \wedge (A \Rightarrow B_{n+1}) \rightarrow (A \Rightarrow B))$
And so, if $(B_1 \wedge \dots \wedge B_n \wedge B_{n+1}) \rightarrow B$ then $((A \Rightarrow B_1) \wedge \dots \wedge (A \Rightarrow B_n) \wedge (A \Rightarrow B_{n+1}) \rightarrow (A \Rightarrow B))$. Which is RCK for $n+1$.

We have seen that RCK is true for $n=1$, so, by induction, we obtain that it is true for any n .

$$\vdash_{S4} \square A \leftrightarrow (\neg A \Rightarrow A).$$

- PC (1) $A \rightarrow A \vee B$
- (1), S4 (2) $\square A \rightarrow \square(A \vee B)$
- (2) $[\square(\neg A \wedge (\neg A \Rightarrow A))]B$
- (3) $\square A \rightarrow \square(A \vee \square(\neg A \wedge (\neg A \Rightarrow A)))$
- (3),Def \Rightarrow (4) $\square A \rightarrow (\neg A \Rightarrow A)$
- PC (5) $A \vee \neg(A \vee A)$
- S4 (6) $\neg(A \vee A) \rightarrow \square(A \vee A)$
- (6),PC (7) $(A \vee \neg(A \vee A)) \rightarrow (A \vee \square(A \vee A))$
- (7),(5),MP (8) $A \vee \square(A \vee A)$
- (8),N (9) $\square(A \vee \square(A \vee A))$
- PC (10) $A \rightarrow (B \rightarrow (B \wedge A))$
- (10)[$\square(A \vee \square(A \vee A))$]A ; $\neg A$ IB (11) $\square(A \vee \square(A \vee A)) \rightarrow (\neg A \rightarrow (\neg A \wedge \square(A \vee \square(A \vee A))))$
- (11),(9),MP (12) $\neg A \rightarrow (\neg A \wedge \square(A \vee \square(A \vee A)))$
- (12),S4 (13) $\neg A \rightarrow \square(\neg A \wedge \square(A \vee \square(A \vee A)))$
- (13),PC (Contrapositive), \neg -Definition. (14) $\square(A \vee \square(\neg A \wedge \square(\neg A \Rightarrow A))) \rightarrow \square A$
- (14),Def \Rightarrow (15) $(\neg A \Rightarrow A) \rightarrow \square A$
- (4),(15),PC (16) $\square A \leftrightarrow (\neg A \Rightarrow A)$

Proof of theorem 6.1:

To do these proofs, we are going to use C4.

Some of results are trivials considering C4-axioms:

Reflexivity is axiom AC5. Rules "Cut", "Cautious Monotonicity", "And" and "Or" are directly obtained considering respectively: AC8, AC9, AC6 and AC7.

$$\frac{\begin{array}{c} A \leftrightarrow B \quad A \Rightarrow C \\ \hline B \Rightarrow C \end{array}}{} \text{Left Logical Equivalence}$$

Let's assume $A \leftrightarrow B$ and $A \Rightarrow C$.

From the first hypothesis, using RCEA, we obtain: $(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)$. Now, considering the second hypothesis and using MP, we can deduce $B \Rightarrow C$.

So, if $A \leftrightarrow B$ and $A \Rightarrow C$ then $B \Rightarrow C$.

$$\frac{\begin{array}{c} A \rightarrow B \quad C \Rightarrow A \\ \hline C \Rightarrow B \end{array}}{} \text{Right Weakening}$$

Let's assume $A \rightarrow B$ and $C \Rightarrow A$.

Consider the first assumption: $A \rightarrow B$. Then $|A| \subseteq |B|$.

Now consider the second assumption: for every C4-model M and any world w of this model: $f(w, |C|) \subseteq |A|$.

Putting together these two results, we obtain: $f(w, |C|) \subseteq |B|$ i.e. $C \Rightarrow B$.

So if $A \rightarrow B$ and $C \Rightarrow A$ then $C \Rightarrow B$.

_ $((A \Rightarrow B) \wedge (B \Rightarrow A) \wedge (A \Rightarrow C)) \rightarrow (B \Rightarrow C)$;

This rule, as Left Logical Equivalence, says that $A \Rightarrow B$ does not depend on the form of A. The only difference is that, instead of material (or classical) equivalence we consider in preferential equivalence.

Let's assume that M is a C4-model and w a world of this model. Let's assume also that $\models_M^*(A \Rightarrow B)$, $\models_M^*(B \Rightarrow A)$ and $\models_M^*(A \Rightarrow C)$.

By definition $\models_M^*(A \Rightarrow B)$ iff $f(w, |A|) \subseteq |B|$. Now considering properties f8 and f9 of the selection function: $f(w, |A|) = f(w, |A \wedge B|)$.

In the same way if $\models_M^* B \Rightarrow A$ then $f(w, |B|) = f(w, |A \wedge B|)$. Then, if both are true, as assumed, we obtain $f(w, |A|) = f(w, |B|)$. And then if $f(w, |A|) \subseteq |C|$ then $f(w, |B|) \subseteq |C|$. I.e. if $\models_M^*(A \Rightarrow C)$ then $\models_M^*(B \Rightarrow C)$.

So we obtain $\models_M^*((A \Rightarrow B) \wedge (B \Rightarrow A) \wedge (A \Rightarrow C)) \rightarrow (B \Rightarrow C)$. This is true for every world in every model. So it is a theorem. The inference rule "equivalence" is directly obtained using this theorem.

_ $(A \wedge B \Rightarrow C) \rightarrow (A \Rightarrow B \Rightarrow C)$.

Let's assume M to be a C4-model and w a world of this model. $\models_M^* A \wedge B \Rightarrow C$ iff $f(w, |A \wedge B|) \subseteq |C|$. And then $f(w, |A \wedge B|) \subseteq |C| \cup \neg |B|$

We also have by f5 that $f(w, |A \wedge \neg B|) \subseteq |A \wedge \neg B|$ and then $f(w, |A \wedge \neg B|) \subseteq |A| \cap \neg |B| \subseteq \neg |B| \subseteq \neg |B| \cup |C|$.

And by f7: $f(w, |(A \wedge \neg B) \vee (A \wedge B)|) \subseteq f(w, |A \wedge \neg B|) \cup f(w, |A \wedge B|)$. Furthermore $((A \wedge B) \vee (A \wedge \neg B)) \leftrightarrow A$ and so $|A \wedge B| \vee |A \wedge \neg B| = |A|$ and in consequence: $f(w, |(A \wedge \neg B) \vee (A \wedge B)|) = f(w, |A|)$.

Putting together these three results: $f(w, |A|) \subseteq \neg |B| \cup |C|$ and so $f(w, |A|) \subseteq |B \Rightarrow C|$.

Conclusion: for any world of any C4-model if $f(w, |A \wedge B|) \subseteq |C|$ then $f(w, |A|) \subseteq |B \Rightarrow C|$.

Hence: $(A \wedge B \Rightarrow C) \rightarrow (A \Rightarrow B \Rightarrow C)$ is a theorem and the inference rule "S" is directly obtained.

_ $((A \wedge \neg B \Rightarrow C) \wedge (A \wedge B \Rightarrow C)) \rightarrow (A \Rightarrow C)$

Let's assume that M is a C4-model and w a world of this model. Like in the previous proof, using f7: $f(w, |A|) \subseteq f(w, |(A \wedge B)|) \cup f(w, |(A \wedge \neg B)|)$. So if $f(w, |(A \wedge B)|) \subseteq |C|$ and $f(w, |(A \wedge \neg B)|) \subseteq |C|$ then $f(w, |A|) \subseteq |C|$.

So: $((A \wedge \neg B \Rightarrow C) \wedge (A \wedge B \Rightarrow C)) \rightarrow (A \Rightarrow C)$ is a theorem, and the inference rule "D" is directly obtained.

References

[Anderson & Nuel & Belnap 1975] Anderson A. R., Nuel D. Belnap JR. "Entailment, the Logic of Relevance and Necessity", Princeton University Press (1975) p594-599.

[Bell 90] Bell J. : "The Logic of Non-Monotonicity", Artificial Intelligence 41, (1989/90) p365-374.

[Boutilier 90] Boutilier Graig : "Conditional Logics of Normality as Modal Systems", Proceedings of Eight National Conference on Artificial Intelligence (AAAI 90) AAAI Press/ MIT Press (1990).

[Brown & Shoham 88] Brown Allen L. Jr., Shoham Yoav: "New results on Semantical Nonmonotonic Reasoning", Lecture Notes in Artificial Intelligence 346, Springer-Verlag, M. Reinfrank J. de Kleer, M.L. Ginsberg, E. Sandewall (eds) (1988) p19-26.

[Chellas 75] Chellas Brian F. "Basic Conditional Logics", Journal of Philosophical Logic 4, D. Reidel Publishing Company, Dordrecht-Holland, (1975), p133-153.

[Delgrande 87] Delgrande James P. : "A First-Order Conditional Logic for Prototypical Properties", Artificial Intelligence 33 (1987), p105-130.

[Delgrande 88] Delgrande James P. : "An Approach to Default Reasoning Based on First Order Conditional Logic", Artificial Intelligence 36 (1988), p63-90.

[Dunn 86] Dunn J. Michael : "Relevant Logic and Entailment", D. Gabbay and F. Guenther (eds), Handbook of Philosophical Logic, Vol III (1986), p117-224.

[Hughes & Cresswell 68] Hughes G., Cresswell M. : "An Introduction to Modal Logic", Methuen Londres, 1968.

[Hughes & Cresswell 84] Hughes G., Cresswell M. : "A companion to Modal Logic", Methuen Londres, 1984.

[Krauss & Lehmann & Magidor 90] Krauss Sarit, Lehmann Daniel, Magidor Menachem : "Nonmonotonic Reasoning, Preferential Models and Cumulative Logics". Artificial Intelligence 44 (1990) p167-207.

[Lamarre 90] Lamarre Philippe : "Modal Logics as Conditional logics of Nonmonotonicity", Internal report IRIT/90-50-R, Université P. Sabatier, Toulouse, France (1990).

[Lehmann 89] Lehmann Daniel : "What does a Conditional Knowledge Base Entail?", Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning, Toronto, Ontario, Canada, Edited by R.J. Brachman, H.J. Levesque, R. Reiter, Morgan Kaufmann Publishers, Inc, San Mateo, California (1989) p212-222.

[Mc Carthy 80] Mc Carthy J. : "Circumscription - A Form of Non-Monotonic Reasoning", Artificial Intelligence 13 (1980) p27-39.

[McDermott & Doyle 80] McDermott D., Doyle J. : "Non-monotonic Logic I", Artificial Intelligence 13 (1980) p41-72.

[McDermott 82] McDermott D. : "Non-monotonic Logic II", Journal of Computer Machinery, Vol. 29. 1. (January

1982), p33-57.

[Makinson 88] Makinson David : "General theory of Cumulative Inference", Non-Monotonic Reasoning, Lecture Notes in Artificial Intelligence 346, Springer-Verlag, M. Reinfrank J. de Kleer, M.L. Ginsberg, E. Sandewall (eds) (1988) p1-18.

[Moore 85] Moore R.C. : "Semantical considerations on Nonmonotonic Logic", Artificial Intelligence 25 (1985) p75-94.

[Nute ??] Nute D. : "A Non-Monotonic logic Based on Conditional Logic", ACM Research Report 01-0007, Advanced Computational Methods Center, University of Georgia,Athen, Georgia.

[Nute 84] Nute D. : "Conditional logics", D. Gabbay and F. Guenther (eds), Philosophical Logic, Vol II, 387-439 1984 by D.J Reidel Publishing company.

[Reiter 80] Reiter R. : "A Logic for Default Reasoning", Artificial Intelligence 13 (1980) p81-132.

[Reiter 87] Reiter R. : "Nonmonotonic Reasoning", Annual Revue of Computer Science-Vol 2 : (1987) p147-86.

[Shoham 87a] Shoham Yoav : "A Semantical Approach to Nonmonotonic Logics", (extended abstract), in Proceedings Logics in Computer Science, Ithaca, NY (1987) p411-435.

[Shoham 87b] Shoham Yoav : "Nonmonotonic Logics : Meaning and Utility" , Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI), Milan, (August 1987), p23-28.

[Stalnaker 68] Stalnaker R. : "A Theory of conditionals", Studies in Logical Theory, N. Rescher (ed.), Basil Blackwell, Oxford, 1968, p41-55.

On the Relation between Explicit and Implicit Belief

Gerhard Lakemeyer*
 Institut für Informatik III
 Universität Bonn
 Römerstraße 164
 W-5300 Bonn 1
 Germany

Abstract

Logics of explicit and implicit belief have been proposed to account for the fact that agents generally believe only some but not all of the logical consequences of their initial beliefs. While the explicit beliefs refer to the beliefs an agent actually holds given his limited reasoning capabilities, the implicit beliefs can be thought of as those the agent would hold if he were an ideally rational agent with unlimited resources. Existing logics of explicit and implicit belief exhibit only a weak connection between the two notions in the sense that, while the explicit beliefs form a subset of the implicit beliefs, there may well be implicit beliefs independent of the explicit ones. In this paper we argue in favor of a stronger connection that is more appropriate in KR. In particular, we want to eliminate extraneous implicit beliefs by requiring that all the implicit beliefs are logical consequences of an agent's explicit beliefs. We demonstrate this idea by appropriately modifying a generalization of an existing logic of explicit and implicit belief.

1 Introduction

It is widely accepted that agents, because of their limited resources, believe some but by no means all of the logical consequences of their initial beliefs.¹ Another way to characterize this situation, as first proposed by Levesque [Levesque 1984], is to distinguish between the *explicit* and *implicit beliefs* of an agent. While the explicit beliefs refer to the beliefs an agent actually

holds given his limited reasoning capabilities, the implicit beliefs can be thought of as those the agent would hold if he were an ideally rational agent with unlimited resources.

In [Levesque 1984], Levesque proposed a logic that deals with both explicit and implicit belief, which allows to compare both notions within a single framework. Other approaches include [Fagin and Halpern 1985, Lakemeyer 1986, 1987].

While there is little agreement on what constitutes an appropriate model of explicit belief, possible-world semantics [Kripke 1963, Hintikka 1962] is generally found to be an adequate model of implicit belief.²

The logics of explicit and implicit belief proposed so far provide a natural connection between the two notions in that the explicit beliefs are always a subset of the implicit beliefs. More precisely, it is a property of these logics that, whenever a sentence is explicitly believed, it follows logically that it is also implicitly believed.

In a sense, this connection alone is rather weak because it does not rule out that an agent holds implicit beliefs that are totally unrelated to his explicit beliefs. While this may be acceptable for an extremely weak reasoner,³ we would like to argue that a stronger connection is called for in a KR setting, where we are interested in modelling the beliefs of a knowledge base (KB). Intuitively, a KB completely determines (represents) all the beliefs of an agent (be they explicit or implicit). If we assume that an agent is minimally rational in the sense that he explicitly believes at least the sentences stored in the KB, we should expect that all of the agent's implicit beliefs *follow* from his explicit beliefs. After all, since the explicit beliefs include the KB, there is no other source of information for the

²In particular, beliefs under possible-world semantics are closed under logical consequence.

³One can imagine a stupid agent who has no explicit beliefs no matter how much information he is given. On the other hand, his implicit beliefs may vary widely, depending on the information given to him. (Joe Halpern suggested this example to me.)

*This work was conducted while the author was at the University of Toronto.

¹For concreteness, one may think of the initial beliefs as consisting of those sentences that are stored in a knowledge base.

implicit beliefs.

In a logic, this strong connection between the explicit and implicit beliefs translates into the property that, given an agent's *epistemic state*, which consists of the beliefs that hold for a particular interpretation of the sentences of the logic, any implicit belief at that state is a *logical consequence* of his explicit beliefs at that state.

Unfortunately, none of the existing approaches has this property, that is, they all admit epistemic states which include implicit beliefs that are independent of their explicit beliefs.

In this paper we show how this problem can be overcome. For that purpose, we first introduce a generalized version of Levesque's logic⁴ [Levesque 1984] in Section 2 and demonstrate that it leads to unwanted epistemic states. In Section 3, we then show how to modify the semantics to rule out the unwanted epistemic states.

2 The Logic *BL*

We consider a propositional modal logic with two modal operators **B** and **L** for explicit and implicit belief, respectively. We begin by describing the two notions informally.

Both forms of belief are defined in a possible-world fashion. Intuitively, we should think of an agent who imagines a set of states of affairs or *situations* *M*, which are an extension of classical (Kripke) worlds and which are described in more detail below. The agent is then said to explicitly believe a sentence α just in case α holds in all situations in *M*. He implicitly believes α just in case α holds in all those situations in *M* that are classical worlds.

A key difference between the implicit and explicit beliefs is that, while the former are closed under *modus ponens*, the latter are not. For example, if *p*, *q*, and *r* are atomic facts, we allow an agent to explicitly believe $(p \vee q) \wedge (\neg q \vee r)$, yet fail to explicitly believe $(p \vee r)$.

In *BL*, closure under modus ponens is avoided by decoupling the notions of truth and falsity within situations. Instead of assigning either true or false to atoms, situations are allowed to assign independent *true-support* and *false-support* to them. This corresponds to using four truth values $\{\}$, $\{\text{true}\}$, $\{\text{false}\}$, and $\{\text{true}, \text{false}\}$, an idea originally proposed to provide a semantics for a fragment of relevance logic called *tautological entailment* [Belnap 1977, Dunn 1976]. Note that the classical worlds of possible-world semantics

are a special kind of situations, namely those where each atomic fact has either true- or false-support but not both. In *BL*, classical worlds are used to provide the standard notions of *truth*, *logical consequence*, and implicit belief. Non-classical situations are only used to determine the explicit beliefs.

2.1 Language and Semantics

The language \mathcal{L} we are considering is a standard propositional language augmented by modal operators **B** and **L** for explicit and implicit belief, respectively.

Definition 1 *The language \mathcal{L}*

*Let \mathcal{P} be a countably infinite set of atomic propositions, which includes the special atom *true*.*

*The sentences of \mathcal{L} consist of the atoms and are closed under \neg , \vee , and the modal operators **L** and **B** except that no **L** may occur inside a **B**.*⁵

Other logical connectives are introduced as usual, that is $\alpha \wedge \beta$, $\alpha \supset \beta$, and $\alpha \equiv \beta$ are abbreviations for $\neg(\neg\alpha \vee \neg\beta)$, $\neg\alpha \vee \beta$, and $(\alpha \supset \beta) \wedge (\beta \supset \alpha)$, respectively. We also write *false* instead of *¬true*. A sentence is called *objective* if it does not contain any modal operator, *basic* if it does not contain any occurrences of **L**, and *subjective* if every atom occurs within the scope of a modal operator.

A situation is completely characterized by the atoms that have true-support and those that have false-support. Worlds, of the ordinary Kripke variety, are situations where every atom has either true- or false-support but not both. Formally:

Definition 2 *A situation s is a tuple $\langle T, F \rangle$, where T and F are subsets of \mathcal{P} such that *true* $\in T$ and *true* $\notin F$.*

(T and F determine the true- and false-support of atoms at the situation.) A situation is called a world if T and F partition \mathcal{P} , that is, every atom is either true or false, but not both.

Given a set of situations *M* and a situation $s = \langle T_s, F_s \rangle$, we can now define the true-support (\models_T) and false-support (\models_F) of any sentence in \mathcal{L} . Let $p \in \mathcal{P}$ and let α and β be arbitrary sentences except

⁴The generalization involves allowing nested beliefs. The notion of explicit belief in this paper is also closely related to the one in [Lakemeyer and Levesque 1988], where it is shown that reasoning based on this model of belief is tractable.

⁵See [Fagin and Halpern 1985] for a logic without this restriction.

in rule 4., where α contains no L's.

1. $M, s \models_T p \iff p \in T_s$
2. $M, s \models_F p \iff p \in F_s$
3. $M, s \models_T \neg\alpha \iff M, s \models_F \alpha$
4. $M, s \models_F \neg\alpha \iff M, s \models_T \alpha$
5. $M, s \models_T \alpha \vee \beta \iff M, s \models_T \alpha \text{ or } M, s \models_T \beta$
6. $M, s \models_F \alpha \vee \beta \iff M, s \models_F \alpha \text{ and } M, s \models_F \beta$
7. $M, s \models_T B\alpha \iff \text{for all } t, \text{ if } t \in M \text{ then } M, t \models_T \alpha$
8. $M, s \models_F B\alpha \iff M, s \models_T B\alpha$
9. $M, s \models_T L\alpha \iff \text{for all worlds } w, \text{ if } w \in M \text{ then } M, w \models_T \alpha$
10. $M, s \models_F L\alpha \iff M, s \models_T L\alpha$

The notions of *truth*, *satisfiability*, and *logical consequence* are defined with respect to worlds and sets of situations: we say that a sentence α is *true* at a world w and a set of situations M if $M, w \models_T \alpha$, otherwise α is *false*. Given a set of sentences Γ and a sentence α , we say that α is a *logical consequence* of Γ ($\Gamma \models \alpha$) iff for all sets of situations M and worlds w , if $M, w \models_T \gamma$ for all $\gamma \in \Gamma$, then $M, w \models_T \alpha$. A sentence α is *valid* ($\models \alpha$) iff α is a logical consequence of the empty set. Finally, a set of sentences Γ is *satisfiable* iff there is a world w and a set of situations M such that every γ in Γ is true at M and w .

Notation: It is easily seen that the support of a subjective sentence at a situation s and a set of situations M is independent of s . Similarly, the support of an objective sentence is independent of M . For that reason, given a subjective sentence σ , we write $M \models_T \sigma$ as shorthand for $M, s \models_T \sigma$ for any situation s . Similarly, for objective sentences α we write $s \models_T \alpha$ instead of $M, s \models_T \alpha$. Given a set of sentences Γ , a set of situations M , and a situation s , we write $M, s \models_T \Gamma$ instead of for all $\gamma \in \Gamma$, $M, s \models_T \gamma$.

2.1.1 Properties of Belief

Since truth, validity, and logical consequence are defined with respect to *worlds* and sets of situations, it is easy to see that the part of the logic that deals only with objective sentences reduces to classical propositional logic.

Implicit belief has the properties of the well known modal logic *weak S5*, which follows from the fact that implicit belief is interpreted with respect to a globally accessible set of worlds (see, for example, [Halpern and Moses 1985]). In particular, implicit beliefs are closed under modus ponens ($\models L\alpha \wedge L(\alpha \supset \beta) \supset L\beta$) and positive as well as negative introspection ($\models L\alpha \supset LL\alpha$ and $\models \neg L\alpha \supset L\neg\alpha$).

We now briefly discuss the properties of explicit belief. (More details can be found in [Lakemeyer 1990]. Also, the special case of explicit belief without nested occurrences of B is treated in [Levesque 1984]. Finally, a version of B that is slightly more com-

plicated than the one in this paper⁶ is presented in [Lakemeyer and Levesque 1988].) Explicit belief is much weaker than implicit belief. The main limitations can be characterized as follows. Let p and q be distinct atoms.

No modus ponens

$$\not\models Bp \wedge B(p \supset q) \supset Bq$$

Tautologies are not necessarily believed

$$\not\models B(p \vee \neg p)$$

Inconsistent beliefs do not imply everything

$$\not\models B(p \wedge \neg p) \supset Bq$$

Apart from these limitations, B shares many properties with the more familiar L . Here are some examples. Let α and β be arbitrary basic sentences.

$$\models B\alpha \supset B(\alpha \vee \beta)$$

$$\models B(\alpha \wedge \beta) \equiv B\alpha \wedge B\beta$$

$$\models B\alpha \supset BB\alpha$$

$$\models \neg B\alpha \supset B\neg B\alpha$$

Note that, as far as introspection is concerned, B is no different from L . The usual equivalence transformations of classical logic such as associativity or De-Morgan's law apply to explicit belief as well. For that reason, arbitrary sentences in BL can always be converted into an equivalent conjunctive normal form, a property that we will make use of below.

Definition 3 Conjunctive Normal Form

A sentence α is in *conjunctive normal form (CNF)* if α is a conjunction of clauses, where a clause is a disjunction of literals and sentences of the form $B\alpha$, $\neg B\alpha$, $L\alpha$, or $\neg L\alpha$ and α is itself in CNF.

An example sentence in CNF is

$$(p \vee Bp) \wedge (Bq \vee \neg L(r \wedge \neg B(t \vee u)) \vee u).$$

We obtain that for any sentence α there is a sentence α_{CNF} such that $\models \alpha \equiv \alpha_{CNF}$.

Similar to *weak S5*, BL has the property that nested beliefs reduce to sentences without nested beliefs. To prove this, we need the following lemma.

Lemma 2.1 Let ρ be a basic subjective sentence, σ a subjective sentence, and γ an objective sentence. Then

$$1. \models B(\rho \vee \gamma) \equiv (\rho \vee B\gamma)$$

$$2. \models L(\sigma \vee \gamma) \equiv (\sigma \vee L\gamma)$$

Proof : Here we only prove 1. The proof of 2. is essentially the same except that we use worlds rather than arbitrary situations.

⁶The complication has to do with an additional concept of "awareness," which we did not include here in order to keep matters simple.

Let M be a set of situations such that $M \models_T B(\rho \vee \gamma)$. Then for all $s \in M$, $M, s \models_T \rho \vee \gamma$. If for all $s \in M$, $M, s \models_T \gamma$, then $M \models_T B\gamma$ and, therefore, $M \models_T \rho \vee B\gamma$. Otherwise, let $s \not\models_T \gamma$ for some $s \in M$. Then $M, s \models_T \rho$, that is, $M \models_T \rho$ since ρ is subjective. Therefore, $M \models_T \rho \vee B\gamma$.

Conversely, let $M \models_T \rho \vee B\gamma$. If $M \models_T B\gamma$, then $M \models_T B(\rho \vee \gamma)$. If $M \models_T \rho$, then for all $s \in M$, $M, s \models_T \rho$, that is, $M \models_T B\rho$, which implies $M \models_T B(\rho \vee \gamma)$. \square

Theorem 1 Reduction Theorem

Let α be a basic sentence and β any sentence. Then there is a basic sentence α' without nested occurrences of B and a sentence β' without nested occurrences of either B or L such that

1. $\models B\alpha \equiv \alpha'$.
2. $\models L\beta \equiv \beta'$.

Proof : First we define the *depth* of a sentence as the largest number of nested modal operators within the sentence. Formally, let $\text{depth}(\alpha) = 0$ for all objective sentences. For arbitrary sentences α and β , let $\text{depth}(\neg\alpha) = \text{depth}(\alpha)$, $\text{depth}(\alpha \vee \beta) = \max\{\text{depth}(\alpha), \text{depth}(\beta)\}$, and $\text{depth}(B\alpha) = \text{depth}(L\alpha) = \text{depth}(\alpha) + 1$. Given this definition, it suffices to show that there are α' and β' of depth at most 1.

1. The proof of the theorem is by induction on $\text{depth}(\alpha)$.

If $\text{depth}(\alpha) = 0$ or $\text{depth}(\alpha) = 1$, the theorem holds trivially with $\alpha = \alpha'$.

Assume the theorem holds for sentences α with $\text{depth}(\alpha) < n$, where $n \geq 2$.

Now let $\text{depth}(\alpha) = n$.

Without loss of generality, we can assume that α is in conjunctive normal form. Thus let $\alpha = \bigwedge \alpha_i$. Since $\models B \bigwedge \alpha_i \equiv \bigwedge B\alpha_i$, it is sufficient to reduce every $B\alpha_i$ to a sentence of depth 1. Let

$$\begin{aligned} \alpha_i &= B\alpha_{i1} \vee \dots \vee B\alpha_{ik-1} \vee \neg B\alpha_{ik} \vee \dots \\ &\quad \vee \neg B\alpha_{il-1} \vee \alpha_{il}, \end{aligned}$$

where α_{il} is a disjunction of literals. If α_{il} does not contain an objective part, that is, if there is no α_{il} , we simply add *false* as an additional disjunct. (Note that $\models B\gamma \equiv B(\gamma \vee \text{false})$.) Now we can apply Lemma 2.1 and obtain

$$\begin{aligned} \models B\alpha_i &\equiv B\alpha_{i1} \vee \dots \vee B\alpha_{ik-1} \vee \neg B\alpha_{ik} \vee \dots \\ &\quad \vee \neg B\alpha_{il-1} \vee B\alpha_{il}. \end{aligned}$$

We repeat this reduction for all α_i and obtain a new formula α^* which has a depth of at most $n-1$. By induction, we can then obtain a formula α' from α^* with depth at most 1.

2. Part 2 is proved the same way. Note that the conjuncts in the conjunctive normal form of β may contain disjuncts of the form $B\alpha$ or $\neg B\alpha$ for basic α . These can be reduced to sentences of depth 1 by applying part 1 of the theorem. \square

2.2 Epistemic States

Among other things, a belief logic gives a precise account of the possible epistemic states of an agent. In the case of *BL*, these are defined by the possible sets of situations.

Definition 4 Belief Sets and Epistemic States

A set of basic sentences Γ_B is called an explicit belief set iff there is a set of situations M such that $\Gamma_B = \{\alpha \mid M \models_T B\alpha\}$. A set of sentences Γ_L is called an implicit belief set iff there is a set of situations M such that $\Gamma_L = \{\alpha \mid M \models_T L\alpha\}$. $\Gamma = \langle \Gamma_B, \Gamma_L \rangle$ is called an epistemic state iff Γ_B and Γ_L are explicit and implicit belief sets, respectively, for the same set of situations M .

Similar to the stable expansions of autoepistemic logic [Moore 1984], epistemic states are uniquely determined by the objective sentences they contain. Let us call the set of objective sentences contained in a set of sentences its *objective subset*.

Theorem 2 Epistemic states are uniquely determined by the objective subsets of their explicit and implicit belief sets.

Proof : Let $\Gamma = \langle \Gamma_B, \Gamma_L \rangle$ and $\Gamma' = \langle \Gamma'_B, \Gamma'_L \rangle$ be epistemic states that agree on the objective subsets of their explicit and implicit beliefs. We need to show that (1.) $\Gamma_B = \Gamma'_B$ and (2.) $\Gamma_L = \Gamma'_L$. Since the proofs of (1.) and (2.) are very similar, we only show (1.) here.

Let $\Gamma_B = \{\alpha \mid M \models_T B\alpha\}$ and $\Gamma'_B = \{\alpha \mid M' \models_T B\alpha\}$. Let α be any basic sentence. By Theorem 1, $\models B\alpha \equiv \alpha^*$ where α^* is subjective and no B occurs nested. To prove $\Gamma_B \subseteq \Gamma'_B$, let $\alpha \in \Gamma_B$. By definition of Γ_B , $M \models_T B\alpha$. Thus, by the reduction theorem, $M \models_T \alpha^*$. Since Γ_B and Γ'_B agree on all objective sentences, a simple induction on the structure of α^* shows that $M' \models_T \alpha^*$, which, in turn, implies $M' \models_T B\alpha$ and, therefore, $\alpha \in \Gamma'_B$. The proof of $\Gamma'_B \subseteq \Gamma_B$ is completely symmetric. \square

Although the explicit and implicit beliefs of an agent are weakly connected, that is, $\models B\alpha \supseteq L\alpha$ for all basic α , the strong connection fails in *BL*, that is, the implicit beliefs of an epistemic state in general are not a logical consequence of its explicit beliefs.

Example 2.1 Let $\Gamma = \langle \Gamma_B, \Gamma_L \rangle$ be an epistemic state for the set of situations

$$M = \{s \mid s \models_T p\} - \{w \mid w \text{ is a world and } w \not\models_T q\}.$$

Clearly, $M \models_T Lq$ and thus $q \in \Gamma_L$. However, $\Gamma_B \not\models q$ as the following counterexample shows.

Let $M' = \{s \mid s \models_T p\}$ with corresponding epistemic state $\Gamma' = \langle \Gamma'_B, \Gamma'_L \rangle$.

First, we show that M and M' agree on their explicit beliefs, i.e. $\Gamma_B = \Gamma'_B$. By the previous theorem, it suffices to show that both sets contain the same objective sentences. Since $M \subseteq M'$, every objective sentence in Γ'_B is also in Γ_B . To prove the other direction, assume, to the contrary, that there is an objective sentence γ such that $M \models_T B\gamma$ and $M' \not\models_T B\gamma$. Thus there is a situation $s \in M'$ such that $s \not\models_T \gamma$. Let r be an atom that does not occur in γ and that is distinct from p and q . Let s^* be a situation which is exactly like s except that $s^* \models_T r \wedge \neg r$. Since γ is objective, $s^* \not\models_T \gamma$ clearly holds. Also $s^* \models_T p$ because $s \in M'$, and s^* is not a world because $s^* \models_T r \wedge \neg r$. Therefore, by the definition of M , $s^* \in M$, which implies that $M \not\models_T B\gamma$ contradicting our initial assumption. Thus M and M' agree on their objective beliefs and, hence, they agree on all explicit beliefs (Theorem 1).

In other words, $M', s \models_T \Gamma_B$ for all $s \in M'$. Let w be a world in M' such that $w \not\models_T q$. Then $M', w \models_T \Gamma_B$ yet $M', w \not\models_T q$. Therefore, $\Gamma_B \not\models q$. This concludes our example.

3 The Logic BL^+

As we have seen, the problem with BL is that it allows for epistemic states whose implicit beliefs are totally unrelated to their explicit beliefs. Our goal is now to eliminate just those unwanted epistemic states without changing the meaning of explicit and implicit belief or, more precisely, without changing the valid sentences of the logic.

3.1 Maximal Sets

We eliminate the unwanted epistemic states by extending the corresponding sets of situations in the following way. Let M be such a set of situations and suppose an objective⁷ γ is implicitly believed, yet γ does not follow logically from the explicit beliefs Γ . Then there must be a world w^* where all objective sentences in Γ are true and where γ is false. If we add w^* to M , the resulting set of situations has the same explicit beliefs as before, namely Γ , but does not implicitly believe γ .

In the following definition of M^+ , we not only add worlds such as the above w^* to M but indeed all situations which support all objective sentences in Γ and which are not already in M . In a sense, M^+ is *maximal* because we cannot add any more situations to M^+ without losing any of the explicit beliefs in Γ .

⁷Note that, by Theorem 1, we only need to consider objective sentences.

Definition 5 Maximal Sets

Given a set of situations M , let

$$M^+ = \{s \mid \text{for all objective } \gamma, M, s \models_T B\gamma \supset \gamma\}.$$

A set of situations M is called maximal iff $M = M^+$.

Examples of maximal sets are $\{s \mid s \models_T \gamma\}$ for any objective sentence γ and the empty set of situations. As an aside, it is worth pointing out that maximal sets are useful for reasons other than properly relating explicit and implicit belief [Lakemeyer and Levesque 1988, Levesque 1990]. The important properties of M^+ are that its explicit beliefs are precisely those of M (Theorem 3) and that it is the largest set with that property (Theorem 4). First, it is not hard to see that M^+ contains M .

Lemma 3.1 For any set of situations M , $M \subseteq M^+$.

Proof : Let $s \in M$ and let γ be any objective sentence. We need to show that $M, s \models_T B\gamma \supset \gamma$. If $s \models_T \gamma$ then $M, s \models_T B\gamma \supset \gamma$. (Note that $B\gamma \supset \gamma$ is the same as $\neg B\gamma \vee \beta$.) Otherwise, if $s \not\models_T \gamma$ then $M \not\models_T B\gamma$ or, equivalently, $M \models_F \neg B\gamma$. Thus $M, s \models_T B\gamma \supset \gamma$ holds as well. \square

Next, we show that M and M^+ are indistinguishable as far as their respective explicit belief sets are concerned.

Theorem 3 For any set of situations M , M and M^+ agree on all explicit beliefs.⁸

Proof : Let Γ and Γ^+ be the explicit belief sets of M and M^+ , respectively. By Theorem 1, it suffices to show that Γ and Γ^+ agree on all objective sentences.

Since $M \subseteq M^+$, the objective subset of Γ^+ is contained in Γ .

To prove the converse, let γ be an objective sentence in Γ , that is, $M \models_T B\gamma$. Suppose $M^+ \not\models_T B\gamma$. Then there is a situation $s \in M^+$ such that $s \not\models_T \gamma$. Therefore $M, s \not\models_T B\gamma \supset \gamma$ and, hence, $s \notin M^+$, a contradiction. \square

Corollary 3.2 For any set of situations M , M^+ is maximal.

Proof : We need to show that $M^+ = M^{++}$. By Lemma 3.1, we know that $M^+ \subseteq M^{++}$. To show that $M^{++} \subseteq M^+$, let $s \in M^{++}$. Thus for all objective γ , $M^+, s \models_T B\gamma \supset \gamma$. By Theorem 3, M , M^+ , and M^{++} agree on all explicit beliefs. Therefore, for all objective γ , $M, s \models_T B\gamma \supset \gamma$, from which $s \in M^+$ follows. \square

⁸The following two theorems and their corollaries have appeared in a modified form in [Lakemeyer and Levesque 1988].

Theorem 4 Given a maximal set M , every set of situations M' with the same explicit belief set as M is a subset of M .

Proof: Let $s \in M'$ with the same explicit beliefs as M and let γ be an arbitrary objective sentence. If $M \models_T B\gamma$ then $M' \models_T B\gamma$ and, therefore, $s \models_T \gamma$. Thus $M, s \models_T B\gamma \supset \gamma$. Since this holds for any objective γ , $s \in M^+$. Since M is maximal, $M = M^+$ and the theorem follows. \square

Corollary 3.3 Maximal sets of situations M and M' agree on all explicit beliefs iff $M = M'$.

Proof: Follows immediately from Theorem 4. \square

3.2 Truth and Logical Consequence in BL^+

Given the notion of a maximal set, the unwanted epistemic states can be removed from BL simply by restricting the semantics to maximal instead of arbitrary sets of situations. More precisely, we obtain a new logic BL^+ from BL in the following way.

1. The language and true- and false-support relations are the same as in BL .
2. Truth, satisfiability, and logical consequence are defined with respect to worlds and maximal sets of situations only. For example, we say that a sentence α is a logical consequence of a set of sentences Γ in BL^+ ($\Gamma \models^+ \alpha$) iff for all maximal sets of situations M and worlds w , if $M, w \models_T \gamma$ for all $\gamma \in \Gamma$, then $M, w \models_T \alpha$. A sentence α is valid in BL^+ ($\models^+ \alpha$) iff α is a logical consequence of the empty set.

3.3 Explicit Beliefs Determine Implicit Beliefs

In the rest of this section, we will formally prove that we have modified the original logic just enough to achieve our objective. On the one hand, we will show that we have not changed the logic too drastically in that the properties of explicit and implicit belief are the same in BL and BL^+ . On the other hand, we will prove that BL^+ indeed fixes the shortcomings of BL in that it preserves only “good” epistemic states, that is, those whose implicit beliefs depend entirely on the explicit ones.

Lemma 3.4 Let Σ be a finite set of sentences of the form $B\alpha$, $\neg B\alpha$, $L\alpha$, or $\neg L\alpha$, where α is objective. Then Σ is satisfiable in BL iff Σ is satisfiable in BL^+ .

Proof: The if direction is trivial since every BL^+ -model is also a BL -model.

To prove the only if direction, let $\Sigma_B = \{B\alpha \mid B\alpha \in \Sigma\}$, $\Sigma_{\neg B} = \{\neg B\alpha \mid \neg B\alpha \in \Sigma\}$, $\Sigma_L = \{L\alpha \mid L\alpha \in \Sigma\}$,

and $\Sigma_{\neg L} = \{\neg L\alpha \mid \neg L\alpha \in \Sigma\}$. Let Σ be satisfiable in BL . Thus there is a set of situations M such that $M \models_T \Sigma$. Let p be an atom not occurring anywhere in Σ and let

$$\begin{aligned} M_p &= \{s \mid s \text{ is the same as some non-world sit.} \\ &\quad s' \in M \text{ except that } s \models_T p \wedge \neg p\} \\ &\cup \{w \mid w \text{ is the same as some world } w' \in M \\ &\quad \text{except that } w \models_T p.\} \end{aligned}$$

Clearly, $M_p \models_T \Sigma$. In addition, $M_p \models_T B(\neg p \vee \alpha)$ for every $L\alpha \in \Sigma_L$ for the following reason: let $s \in M_p$. If s is not a world, then $s \models_T p \wedge \neg p$ and, therefore, $s \models_T \neg p \vee \alpha$. If s is a world, then, by assumption, $s \models_T \alpha$ and thus $s \models_T \neg p \vee \alpha$. Also, by construction of M_p , $M_p \models_T Bp$.

Now let us consider the maximal set M_p^+ . First, since M_p and M_p^+ agree on all explicit beliefs, $M_p^+ \models_T \Sigma_B$. Also, since $M_p \subseteq M_p^+$, $M_p^+ \models_T \Sigma_{\neg B}$ and $M_p^+ \models_T \Sigma_{\neg L}$. To show that $M_p^+ \models_T \Sigma_L$, let $L\alpha$ be any sentence in Σ_L . Since $M_p \models_T Bp \wedge B(\neg p \vee \alpha)$, $M_p^+ \models_T Bp \wedge B(\neg p \vee \alpha)$ as well. Then $M_p^+ \models_T Lp \wedge L(\neg p \vee \alpha)$ because every explicit belief is implicitly believed. Finally, since implicit beliefs are closed under logical consequence, $M_p^+ \models_T L\alpha$. \square

Theorem 5 BL and BL^+ agree on all valid sentences. For any sentence α , $\models \alpha$ iff $\models^+ \alpha$.

Proof: The only if direction follows immediately because every BL^+ -model is also a BL -model.

Conversely, assume $\not\models \alpha$, i.e., there is a world and a set of situations M such that $M, w \not\models_T \alpha$. We show that there exists a maximal set M' such that $M', w \models_T \neg \alpha$, which is the same as $M', w \not\models_T \alpha$ because w is a world. First, by Theorem 1, there is a sentence α' without nested beliefs such that $\models \neg \alpha \equiv \alpha'$. Thus $\models^+ \neg \alpha \equiv \alpha'$ as well. Let

$$\begin{aligned} \Sigma &= \{B\gamma \mid B\gamma \text{ occurs in } \alpha' \text{ and } M \models_T B\gamma\} \cup \\ &\quad \{\neg B\gamma \mid \neg B\gamma \text{ occurs in } \alpha' \text{ and } M \models_T \neg B\gamma\} \cup \\ &\quad \{L\gamma \mid L\gamma \text{ occurs in } \alpha' \text{ and } M \models_T L\gamma\} \cup \\ &\quad \{\neg L\gamma \mid \neg L\gamma \text{ occurs in } \alpha' \text{ and } M \models_T \neg L\gamma\}. \end{aligned}$$

By Lemma 3.4, there is a maximal set M' such that $M' \models_T \Sigma$. Then, by a simple induction argument on the structure of α' , $M', w \models_T \alpha'$ follows, and, therefore, $M', w \models_T \neg \alpha$. \square

What this theorem says is that, as far as finite sets of sentences are concerned, BL and BL^+ are indistinguishable. In particular, the properties of explicit and implicit belief that are expressible as valid sentences in BL are precisely the same in BL^+ .

This is not the case, however, for infinite sets of sentences such as epistemic states. In particular, the following theorem shows the desired result that, in BL^+ , any implicit belief of an agent is a logical consequence of his explicit beliefs.

Theorem 6 Let M be a maximal set of situations and α any sentence. Let Γ be the explicit belief set of M . Then

$$M \models_T L\alpha \text{ iff } \Gamma \models^+ \alpha.$$

Proof: To prove the if direction, let $\Gamma \models^+ \alpha$. Since $M \models_T B\gamma$ for all $\gamma \in \Gamma$, $M \models_T L\gamma$ holds as well. Thus for all $\gamma \in \Gamma$ and all worlds $w \in M$, $M, w \models_T \gamma$. Since, by assumption, $\Gamma \models^+ \alpha$, $M, w \models_T \alpha$ for all $w \in M$, which immediately implies $M \models_T L\alpha$.

Conversely, assume that $M \models_T L\alpha$. Let M' be a maximal set and w a world such that $M', w \models_T \Gamma$. We need to show that $M', w \models_T \alpha$. To see that M and M' agree on all explicit beliefs, notice that for all $\gamma \in \Gamma$, $B\gamma \in \Gamma$ by positive introspection and thus $M' \models_T B\gamma$. Conversely, if $\gamma \notin \Gamma$, then $\neg B\gamma \in \Gamma$ by negative introspection. Thus $M' \not\models_T B\gamma$.

Thus M and M' agree on all explicit beliefs, from which $M = M'$ follows by Corollary 3.3. Next we show that $w \in M'$. By assumption, $M', w \models_T \Gamma$. Therefore, for all objective γ , if $M' \models_T B\gamma$ (i.e. $\gamma \in \Gamma$), then $w \models_T \gamma$. Thus, by the definition of maximal sets, $w \in M'$.

Since $M = M'$, $M' \models_T L\alpha$ and, since $w \in M'$, $M', w \models_T \alpha$ and we are done. \square

In Theorem 6, the condition for implicitly believing a sentence α is that α is a logical consequence of Γ . We can give an equivalent condition that says that *implicitly believing α* is a logical consequence of *explicitly believing* the sentences in Γ .

Theorem 7 Let M be a maximal set of situations and α any sentence. Let Γ be the explicit belief set of M . Then

$$\Gamma \models^+ \alpha \text{ iff } \{B\gamma \mid \gamma \in \Gamma\} \models^+ L\alpha.$$

Proof: To prove the only if direction, assume $\Gamma \models^+ \alpha$ and let $M \models_T \{B\gamma \mid \gamma \in \Gamma\}$. Then for all worlds $w \in M$, $M, w \models_T \Gamma$, which, together with the assumption, implies $M \models_T L\alpha$.

Conversely, assume $\{B\gamma \mid \gamma \in \Gamma\} \models^+ L\alpha$ and let $M, w \models_T \Gamma$ for some maximal M and world w . Since $M \models_T \{B\gamma \mid \gamma \in \Gamma\}$, $M \models_T L\alpha$ by assumption. Also, for all objective $\gamma \in \Gamma$, $M, w \models_T B\gamma \supset \gamma$. By the definition of maximal sets, $w \in M$. Since $M \models_T L\alpha$, $M, w \models_T \alpha$ follows. \square

Note that the fact that $\Gamma \models^+ \alpha$ and $\{B\gamma \mid \gamma \in \Gamma\} \models^+ L\alpha$ are equivalent conditions is contingent on our notion of implicit belief. For example, if we were to use a weaker notion of implicit belief⁹, Theorem 7 would no longer hold. In a sense, Theorem 6 is a better characterization because it allows us to immediately read off

⁹In the extreme, one could choose L to be the same as B .

important characteristics of implicit belief such as its closure under *modus ponens*.

4 Conclusions

In this paper, we presented a logic of explicit and implicit belief, which captures the idea that all of an agent's implicit beliefs are logical consequences of his explicit beliefs. Such a strong connection between the two notions of belief seems appropriate for knowledge-based agents all of whose beliefs are determined by their KB and who are minimally rational in the sense that they explicitly believe at least the sentences in the KB.

So far, the strong connection between explicit and implicit belief can only be studied in metalogical terms, since it involves infinite sets of sentences (epistemic states). For certain epistemic states, namely those that are *representable* by a sentence (or a finite collection of them), we can do better. In particular, any *objective* sentence γ can be said to represent the epistemic state Γ_γ of the maximal set $M_\gamma = \{s \mid s \models_T \gamma\}$.¹⁰ In [Lakemeyer 1990], we add a new operator O to the logic,¹¹ where $O\gamma$ may be read as “ γ is all that is explicitly believed,” which allows us to completely characterize Γ_γ within the logic in that

$$M_\gamma \models_T B\alpha \text{ iff } \models O\gamma \supset B\alpha.$$

Going back to the problem addressed in this paper, we are then able to express within the logic whether an implicit belief follows from Γ_γ in the sense that

$$M_\gamma \models_T L\alpha \text{ iff } \models O\gamma \supset L\alpha.$$

For example, in this extended logic $\models O p \supset L q$ and indeed $\models O p \supset \neg L q$, that is, if all you explicitly believe is p then it follows logically that you do not implicitly believe q .

Acknowledgements

I am grateful to Hector Levesque for many fruitful discussions on the subject of explicit and implicit belief. Financial support was received by the Department of Computer Science at the University of Toronto.

References

- [Belnap 1977] Belnap, N. D., A Useful Four-Valued Logic, in G. Epstein and J. M. Dunn (eds.), *Modern Uses of Multiple-Valued Logic*, Reidel, 1977.
- [Dunn 1976] Dunn, J. M., Intuitive Semantics for First-Degree Entailments and Coupled Trees, *Philosophical Studies* 29, 1976, pp. 149–168.

¹⁰In this sense, BL^+ tells us precisely what the beliefs of an objective knowledge base are.

¹¹This notion of “all-I-know” was originally proposed by Levesque [Levesque 1990] for implicit belief.

- [Fagin and Halpern 1985] Fagin, R. and Halpern, J. Y., Belief, Awareness, and Limited Reasoning: Preliminary Report, in *Proc. International Joint Conference on Artificial Intelligence*, Los Angeles, 1985, pp. 491–501.
- [Halpern and Moses 1985] Halpern, J. Y. and Moses, Y. O., A Guide to the Modal Logics of Knowledge and Belief, in *Proc. of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1985, pp. 480–490.
- [Hintikka 1962] Hintikka, J., *Knowledge and Belief: An Introduction to the Logic of the Two Notions*, Cornell University Press, 1962.
- [Kripke 1963] Kripke, S. A., Semantical Considerations on Modal Logic, *Acta Philosophica Fennica* 16, 1963, pp. 83–94.
- [Lakemeyer 1986] Lakemeyer, G., Steps Towards a First-Order Logic of Explicit and Implicit Belief, in *Proc. of the Conference on Theoretical Aspects of Reasoning about Knowledge*, Asilomar, California, 1986, pp. 325–340.
- [Lakemeyer 1987] Lakemeyer, G., Tractable Meta-Reasoning in Propositional Logics of Belief, in *Proc. of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, 1987, pp. 402–408.
- [Lakemeyer and Levesque 1988] Lakemeyer, G. and Levesque, H. J., A Tractable Knowledge Representation Service with Full Introspection, in *Proc. of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, Asilomar, California, 1988, pp. 145–159.
- [Lakemeyer 1990] Lakemeyer, G., *Models of Belief for Decidable Reasoning in Incomplete Knowledge Bases*, Ph.D. thesis, Department of Computer Science, University of Toronto, 1990.
- [Levesque 1984] Levesque, H. J., A Logic of Implicit and Explicit Belief, Tech. Rep. No. 32, Fairchild Lab. for AI Research, Palo Alto, 1984.
- [Levesque 1990] Levesque, H. J., All I Know: A Study in Autoepistemic Logic, *Artificial Intelligence*, North Holland, 42, 1990, pp. 263–309.
- [Moore 1984] Moore, R., Possible World Semantics for Autoepistemic Logic, in *The Non-Monotonic Reasoning Workshop*, New Paltz, NY, pp. 344–354.

Toward a Metatheory of Action

Vladimir Lifschitz
 Department of Computer Sciences
 and Department of Philosophy
 University of Texas at Austin
 Austin, TX 78712

Abstract

We present a formalization of the blocks world on the basis of the situation calculus and circumscription, and investigate its mathematical properties. The main theorem describes the effect of the circumscription which solves the frame problem in the presence of ramifications. The theorem is quite general, in the sense that it is applicable not only to the blocks world, but to a whole class of domains involving situations and actions. Its statement does not mention anything specific for the domain of blocks. Instead, it lists assumptions about purely formal, mostly syntactic, properties of the axiom set.

1 INTRODUCTION

In this paper we present a formalization of the blocks world on the basis of the situation calculus and circumscription, and investigate its mathematical properties.

The formalization is not particularly original; it is based on the approach of [Baker, 1989]¹, and is in some ways similar to the formulation from Section 6 of [Baker and Ginsberg, 1989]. One difference is that we use a more abstract formalization of states, along the lines of [Lifschitz, 1990], which allows us, for

¹This was the first successful application of circumscription [McCarthy, 1986] to the frame problem for actions with indirect effects ("ramifications").

instance, to do without any specific assumptions about the number of available blocks.

The main novelty here is a theorem, which describes the effect of the circumscription used for solving the frame problem in the presence of ramifications, and confirms the adequacy of this solution. The theorem is quite general, in the sense that it is applicable not only to the blocks world, but to a whole class of domains involving situations and actions. Its statement does not mention anything specific for the domain of blocks. Instead, it lists assumptions about purely formal, mostly syntactic, properties of the axiom set. One of the assumptions, for instance, is that the axiom set includes a certain form of the commonsense law of inertia.

Such "metatheoretical" investigation of commonsense knowledge gives us not merely one successful formalization of one particular domain, but a class of successful formalizations. This approach has two advantages.

First, any new commonsense domain that we want to formalize may happen to have a formalization which belongs to a class already familiar to us. Then the confirmation of the adequacy of this formalization will come from a metatheorem proved earlier.

Second, databases of common sense, like any other databases, will need to be updated, and it is crucial that an update be allowed only when the designer of the database has a clear understanding of the effect of updates of that type. Hopefully, this can be achieved by implementing the requirement that the updated database belong to a class of theories whose properties are well understood. The metalevel investigation of formalizations of common sense will be instrumental for solving this problem.

The main theorem, stated in Section 3 of this paper and proved in Section 4, shows how these ideas apply to circumscriptive theories of action. Prior to discussing this theorem, we describe in Section 2 a formalization of the blocks world which satisfies its

conditions and thus plays the role of a motivating example. The version of the blocks world used in this example is extremely simple; the only available action is moving one block on top of another. But the theorem covers more complex domains as well. We discuss its possibilities and limitations in Sections 3.3 and 3.4; we show, in particular, that it is applicable to the blocks world in which blocks have colors and can be not only moved, but also painted.

2 THE BLOCKS WORLD EXAMPLE

2.1 CIRCUMSCRIPTIVE THEORIES

The blocks world will be formalized as a *circumscriptive theory*, in the following sense of this term. Consider a language of classical predicate calculus with equality—one-sorted or many-sorted, first-order or higher-order. A circumscriptive theory consists of a finite set Γ of sentences of this language, called the *axioms* of the theory, and a finite nonempty set Δ of expressions of the form

$$\text{circ } P \text{ var } Z_1, \dots, Z_n, \quad (1)$$

where P is a predicate constant, and Z_1, \dots, Z_n are predicate or function constants. These expressions will be called the *policy declarations* of the theory (because they determine its “circumscription policy”). A circumscriptive theory (Γ, Δ) represents the classical axiomatic theory whose axioms are the sentences²

$$\text{CIRC} \left[\bigwedge_{A \in \Gamma} A; P; Z_1, \dots, Z_n \right] \quad (2)$$

for all declarations (1) from Δ . In particular, by a *model* of (Γ, Δ) we mean a model of the formulas (2). A *theorem* of (Γ, Δ) is a sentence which is true in all its models.

The language of a circumscriptive theory can be many-sorted and higher order. In most applications of circumscription, higher order variables do not occur in the axioms, and are only needed for forming the circumscription formula. Baker ([Baker, 1989]), Section 5) noticed, however, that predicate variables are useful for stating his “existence of situations” axioms, and we follow this approach in Section 2.2.

Extending the definition of circumscription to the case when some or all arguments of the circumscribed predicate are higher order variables is straightforward.

²CIRC[A; P; Z] stands for the result of circumscribing the predicate P in the sentence A with Z allowed to vary [McCarthy, 1986], [Lifschitz, 1985], [Genesereth and Nilsson, 1987].

2.2 AXIOMS FOR SITUATIONS AND FLUENTS

First we describe the part of the axiom set which deals with situations and fluents; axioms for actions will be added in Section 2.3. This part of the formalization uses variables of three sorts: for blocks $(x, y, z, x_1, y_1, z_1, \dots)$, for situations (s, s_1, \dots) , and for propositional fluents (f, f_1, \dots) . The formula $\text{Holds}(f, s)$ expresses that the value of f in the situation s is *true*. The specific fluents that are of interest to us in this example will be represented using the function constants *On* and *Ontable*. The terms *On*(x, y) and *Ontable*(x) represent distinct fluents for different values of x and y :

$$\begin{aligned} \text{On}(x_1, y_1) &= \text{On}(x_2, y_2) \supset x_1 = x_2 \wedge y_1 = y_2, \\ \text{Ontable}(x) &= \text{Ontable}(y) \supset x = y, \\ \text{On}(x, y) &\neq \text{Ontable}(z). \end{aligned} \quad (3)$$

The fluents *On*(x, y) and *Ontable*(x) play the role of a “coordinate frame” in the space of situations, in the sense that specific configurations of blocks can be described by combinations of values of these “frame fluents” [McCarthy and Hayes, 1969]. For instance, the configuration in which all blocks are on the table can be characterized by saying that the value of each of the fluents *Ontable*(x) is *true*.³

By a “state” we understand a function assigning values to all frame fluents. When the frame fluents are propositional, a state is a truth-valued function on the set of frame fluents, and we will identify it with the set of fluents to which it assigns the value *true*. Every situation defines a certain state—the set of all frame fluents that are *true* in this situation. But one state may correspond to many different situations. For instance, two different situations s_1 and s_2 may share the property that all blocks are located on the table; perhaps the difference between s_1 and s_2 is in the exact locations of blocks, or these situations may correspond to different instants of time.

To formalize these ideas, we use the unary predicate *Frame*, which singles out the frame fluents.⁴ It is characterized by the postulates:

$$\begin{aligned} \text{Frame}(\text{On}(x, y)), \\ \text{Frame}(\text{Ontable}(x)). \end{aligned} \quad (4)$$

We will circumscribe it, to ensure that the frame contains no fluents other than these.

Since states are sets of frame fluents, they can be represented by unary predicates. We will use σ, σ_1, \dots as unary predicate variables whose argument is a fluent

³See [Lifschitz, 1990] for a detailed discussion of the role of frames and of possible approaches to formalizing this concept.

⁴For a detailed discussion of this approach to formalizing frames, see [Lifschitz, 1990].

variable. We write $IsState(\sigma)$ for

$$\forall f[\sigma(f) \supset Frame(f)],$$

and $State[s]$ for

$$\lambda f[Frame(f) \wedge Holds(f, s)].$$

Obviously, the formula

$$\forall s IsState(State[s])$$

is universally valid.

Given a state σ , we can ask whether there exist a situation s such that $State[s] = \sigma$. Many combinations of values of frame fluents are impossible to realize physically, and even difficult to imagine. A block cannot be located in two places at once; a block cannot be located on its own top; it is impossible to build "circular" configurations, when, for instance, x is located on top of y , y on top of z , and z on top of x . However, we do not think of every situation as necessarily realized at some point in the actual course of events; consequently, there would be nothing wrong with admitting "ideal" situations, corresponding to physically impossible states. In fact, postulating such "ideal" situations is useful, because it allows us to guarantee the existence of a variety of different situations without going into the detailed study of what is physically possible and what is not.

The formula

$$IsState(\sigma) \supset \exists s (\sigma = State[s])$$

is a very strong "existence of situations" axiom; it asserts that every state corresponds to at least one situation, so that the frame fluents are completely independent. Alternatively, this assumption can be stated in the Skolemized form:

$$IsState(\sigma) \supset \sigma = State[Sit(\sigma)]. \quad (5)$$

The function Sit maps a state into one of the corresponding situations.

In our formalization of the blocks world, we do not want to go quite this far in allowing "ideal" situations. We will assume the constraints on possible combinations of the values of frame fluents, according to which a block cannot be located in two places at once:

$$\begin{aligned} Holds(On(x, y_1), s) \wedge Holds(On(x, y_2), s) \\ \supset y_1 = y_2, \end{aligned} \quad (6)$$

$$Holds(On(x, y), s) \supset \neg Holds(Ontable(x), s).$$

Then (5) needs to be replaced by the corresponding "default":

$$IsState(\sigma) \wedge \neg Inconsistent(\sigma) \supset \sigma = State[Sit(\sigma)], \quad (7)$$

where $Inconsistent$ is the new predicate, which will be circumscribed.

We expect that this circumscription will lead to the conclusion that the constraints (6) represent all existing dependencies between the values of frame fluents. One more assumption is needed in order to make this work:

$$\begin{aligned} IsState(\sigma_1) \wedge IsState(\sigma_2) \wedge Sit(\sigma_1) = Sit(\sigma_2) \\ \supset \sigma_1 = \sigma_2. \end{aligned} \quad (8)$$

This axiom guarantees that the cardinality of the universe of situations is sufficiently large.

To sum up, the axioms for situations and fluents are the universal closures of the formulas (3), (4) and (6)–(8).

2.3 AXIOMS FOR ACTIONS

Now we extend the language used in Section 2.2 by variables for actions a, a_1, \dots . We also add the binary function constant $Result$, whose arguments are an action and a situation and whose value is a situation, and the binary function constant $Move$, whose arguments are blocks and whose value is an action. Intuitively, the term $Move(x, y)$ represents the action of placing x on top of y . These terms represent distinct actions for different values of x and y :

$$Move(x_1, y_1) = Move(x_2, y_2) \supset x_1 = x_2 \wedge y_1 = y_2. \quad (9)$$

We will use the atomic formula $Possible(a, s)$ to express that it is possible to carry out the action a in the situation s . The following axiom describes the effect of $Move(x, y)$:

$$\begin{aligned} Possible(Move(x, y), s) \\ \supset Holds(On(x, y), Result(Move(x, y), s)). \end{aligned} \quad (10)$$

A sufficient condition for the possibility of $Move(x, y)$ is given by the axiom:

$$\begin{aligned} \neg \exists z Holds(On(z, x), s) \\ \wedge \neg \exists z Holds(On(z, y), s) \\ \wedge x \neq y \\ \supset Possible(Move(x, y), s). \end{aligned} \quad (11)$$

Finally, we include the following form of the commonsense law of inertia:

$$\begin{aligned} Frame(f) \wedge Possible(a, s) \wedge \neg Noninertial(f, a, s) \\ \supset [Holds(f, Result(a, s)) \equiv Holds(f, s)]. \end{aligned} \quad (12)$$

Here $Noninertial$ is a new predicate constant, which will be circumscribed.

2.4 POLICY DECLARATIONS

By BW we denote the circumscriptive theory whose axioms are the universal closures of the formulas (3), (4) and (6)–(12), and whose policy declarations are:

$$\begin{aligned} \text{circ } Frame \text{ var } Inconsistent, Noninertial, \\ \text{circ } Inconsistent \text{ var } Holds, Result, Possible, \\ \quad Noninertial, \\ \text{circ } Noninertial \text{ var } Result. \end{aligned} \quad (13)$$

Notice that *Inconsistent* is varied when *Frame* is circumscribed, but not the other way around; in this sense, *Frame* is minimized at a higher priority than *Inconsistent*. This is because we want to think of the extent of the frame as already fixed when we decide which states are consistent and which are not. Furthermore, this circumscription policy minimizes *Noninertial* at a lower priority than the other two circumscribed predicates, because we think of the structure of the space of situations as already determined when the effects of actions are described. Since the intention of the law of inertia is to characterize the result of executing a , the function *Result* is varied as *Noninertial* is circumscribed.

2.5 EFFECT OF THE CIRCUMSCRIPTIONS

The main theorem, applied to the theory *BW*, will show that the set of theorems of *BW* includes certain complete characterizations (explicit definitions) of all three circumscribed predicates.

The definition of *Frame* is quite simple:

$$\begin{aligned} \text{Frame}(f) \\ \equiv \exists xy[f = \text{On}(x, y)] \vee \exists x[f = \text{Ontable}(x)]. \end{aligned}$$

This is indeed what we expected to get when we postulated that all fluents of the forms $\text{On}(x, y)$ and $\text{Ontable}(x)$ belong to the frame, and circumscribed *Frame*. Notice that this equivalence is the result of circumscribing *Frame* relative to the conjunction F of (the universal closures of) the axioms (4), so that it can be written in the form

$$\text{CIRC}[F; \text{Frame}]. \quad (14)$$

To describe the effect of circumscribing the predicate *Inconsistent*, we need the following notation: $C_0(\sigma)$ stands for⁵

$$\begin{aligned} \forall xy_1y_2[\sigma(\text{On}(x, y_1)) \wedge \sigma(\text{On}(x, y_2)) \supset y_1 = y_2] \\ \wedge \forall xy[\sigma(\text{On}(x, y)) \supset \neg\sigma(\text{Ontable}(x))]. \end{aligned}$$

This formula expresses the constraint on the state σ similar to the constraint we have imposed on situations: A block cannot be in two places at once. The characterization of *Inconsistent* given by the theory *BW* is:

$$\text{Inconsistent}(\sigma) \equiv \text{IsState}(\sigma) \wedge \neg C_0(\sigma). \quad (15)$$

Thus the state is consistent unless it requires some block to be in two places at once.

For determining the minimal extent of *Noninertial*, we need the following notation: $R_0(a, f)$ stands for

$$\exists xy(a = \text{Move}(x, y) \wedge f = \text{On}(x, y)).$$

⁵The reason why we chose this particular symbol is that it is convenient in the general framework of the main theorem. The same can be said about the symbol R_0 introduced below.

This formula expresses that a “causes” f . By $\text{Affected}(f, a, \sigma)$ (“the fluent f is affected by the action a in the state σ ”) we denote the formula

$$\begin{aligned} \forall\sigma_1[\text{IsState}(\sigma_1) \wedge C_0(\sigma_1) \\ \wedge \forall f_1(R_0(a, f_1) \supset \sigma_1(f_1)) \\ \supset \neg(\sigma_1(f) \equiv \sigma(f))]. \end{aligned}$$

The antecedent of this conditional expresses that σ_1 is a consistent state that can occur after the execution of a . Thus we say that f is affected by a in the state σ if the value of f in any such state σ_1 is different from its value in σ .

Using this notation, we can express the characterization of *Noninertial* given by *BW* by the formula:

$$\begin{aligned} \text{Noninertial}(f, a, s) \equiv \\ \text{Frame}(f) \wedge \text{Possible}(a, s) \wedge \text{Affected}(f, a, \text{State}[s]). \end{aligned} \quad (16)$$

To illustrate the role of this conclusion, we will show that it allows us to prove the “frame axiom”

$$\begin{aligned} \text{Possible}(\text{Move}(x_1, y_1), s) \wedge x_1 \neq x_2 \\ \supset [\text{Holds}(\text{On}(x_2, y_2), \text{Result}(\text{Move}(x_1, y_1), s)) \\ \equiv \text{Holds}(\text{On}(x_2, y_2), s)]. \end{aligned} \quad (17)$$

Define

$$\text{About}(f, x) \equiv f = \text{Ontable}(x) \vee \exists y(f = \text{On}(x, y)).$$

The axioms of *BW* imply

$$\begin{aligned} \text{IsState}(\sigma) \wedge C_0(\sigma) \wedge x_1 \neq x_2 \supset \\ \neg\text{Affected}(\text{On}(x_2, y_2), \text{Move}(x_1, y_1), \sigma), \end{aligned}$$

because, whenever $\text{IsState}(\sigma) \wedge C_0(\sigma) \wedge x_1 \neq x_2$, we can get a counterexample to

$$\text{Affected}(\text{On}(x_2, y_2), \text{Move}(x_1, y_1), \sigma)$$

by taking

$$\sigma_1 = \lambda f[(\sigma(f) \wedge \neg\text{About}(f, x_1)) \vee f = \text{On}(x_1, y_1)].$$

Now, using (6), we conclude:

$$\begin{aligned} x_1 \neq x_2 \\ \supset \neg\text{Affected}(\text{On}(x_2, y_2), \text{Move}(x_1, y_1), \text{State}[s]). \end{aligned}$$

Then, by (16),

$$\begin{aligned} x_1 \neq x_2 \\ \supset \neg\text{Noninertial}(\text{On}(x_2, y_2), \text{Move}(x_1, y_1), s), \end{aligned}$$

and the formula (17) follows from the law of inertia (12).

Another “frame axiom,”

$$\begin{aligned} \text{Possible}(\text{Move}(x_1, y_1), s) \wedge x_1 \neq x_2 \\ \supset [\text{Holds}(\text{Ontable}(x_2), \text{Result}(\text{Move}(x_1, y_1), s)) \\ \equiv \text{Holds}(\text{Ontable}(x_2), s)], \end{aligned}$$

can be proved in a similar way.

3 MAIN THEOREM

Now we want to look at the formalization of the blocks world described above from a more general point of view.

3.1 SETTING THE STAGE FOR THE MAIN THEOREM

We assume a many-sorted language L , containing object variables for situations (s, s_1, \dots), for propositional fluents (f, f_1, \dots), for actions (a, a_1, a_2, \dots), and possibly object variables of other, domain-dependent sorts. L may contain higher order variables. It is assumed to contain variables $\sigma, \sigma_1, \sigma_2, \dots$ for properties of fluents (that is, unary predicates with a fluent argument).

In the blocks world example, there is one domain-dependent sort—blocks.

L is assumed to contain the following function and predicate constants:

- the binary predicate *Holds*, whose arguments are a fluent and a situation,
- the unary predicate *Frame*, whose argument is a fluent,
- the unary predicate *Inconsistent*, whose argument is a property of fluents,
- the binary function *Result*, whose arguments are an action and a situation, and whose value is a situation,
- the binary predicate *Possible*, whose arguments are an action and a situation,
- the ternary predicate *Noninertial*, whose arguments are a fluent, an action and a situation.

These constants will be called *essential*. Besides the essential constants, L is assumed to contain the unary function constant *Sit*, whose argument is a property of fluents, and whose value is a situation.

L may also contain other object, function and predicate constants.

In the blocks world example, there are 3 additional constants: the function constants *On*, *Ontable* and *Move*.

Notice that the abbreviations *IsState* and *State*, introduced in Section 2.2, can be used in any language L satisfying these conditions.

By T we denote a circumscriptive theory in the language L . The axiom set of T may contain any sentences without essential constants—we will call these axioms *inessential*—and it is assumed to contain certain *essential* axioms, described below.

In the blocks world example, the inessential axioms are the universal closures of (3) and (9).

There are 7 essential axioms:

1. F (“the axiom for *Frame*”), which is assumed to contain no essential constants other than *Frame*.

In the blocks world example, F is the conjunction of the universal closures of the formulas (4).

2. S (“the axiom for *Sit*”) is the universal closure of (7), that is,

$$\begin{aligned} \forall \sigma. & IsState(\sigma) \wedge \neg Inconsistent(\sigma) \\ & \supset \sigma = State[Sit(\sigma)]. \end{aligned}$$

3. U (“the uniqueness axiom for *Sit*”) is the universal closure of (8), that is,

$$\begin{aligned} \forall \sigma_1 \sigma_2. & IsState(\sigma_1) \wedge IsState(\sigma_2) \\ & \wedge Sit(\sigma_1) = Sit(\sigma_2) \\ & \supset \sigma_1 = \sigma_2. \end{aligned}$$

4. C (“the domain constraint”) is

$$\forall s. C_0(State[s]),$$

where $C_0(\sigma)$ is a formula containing no essential constants and no free variables other than σ .

In the blocks world example, $C_0(\sigma)$ is selected as in Section 2.5. Let us see what C is for this choice of C_0 . The first conjunctive term of C is (the universal closure of)

$$State[s](On(x, y_1)) \wedge State[s](On(x, y_2)) \supset y_1 = y_2,$$

that is,

$$\begin{aligned} & Frame(On(x, y_1)) \\ & \wedge Holds(On(x, y_1), s) \\ & \wedge Frame(On(x, y_2)) \\ & \wedge Holds(On(x, y_2), s) \\ & \supset y_1 = y_2. \end{aligned}$$

In the presence of (4), this is equivalent to the first of the formulas (6). Similarly, the remaining part of C gives the second of these formulas. Consequently, including C in the axiom set is equivalent to including (6).

5. R (“the axiom for *Result*”) is

$$\begin{aligned} \forall a fs. & Possible(a, s) \wedge R_0(a, f) \\ & \supset Holds(f, Result(a, s)), \end{aligned}$$

where $R_0(a, f)$ is a formula containing no essential constants and no free variables other than a and f .

In the blocks world example, $R_0(a, f)$ is defined as in Section 2.5. In this case, R is equivalent to

$$\begin{aligned} \forall a fs xy. & Possible(a, s) \\ & \wedge a = Move(x, y) \\ & \wedge f = On(x, y) \\ & \supset Holds(f, Result(a, s)), \end{aligned}$$

or

$$\begin{aligned} \forall s xy. & Possible(Move(x, y), s) \\ & \supset Holds(On(x, y), Result(Move(x, y), s)), \end{aligned}$$

which is the universal closure of (10).

6. P ("the axiom for Possible") is the formula

$$\forall s. P_0(a, \text{State}[s]) \supset \text{Possible}(a, s),$$

where $P_0(a, \sigma)$ is a formula containing no essential constants and no free variables other than a and σ .

In the blocks world example, $P_0(a, \sigma)$ is

$$\begin{aligned} \exists xy.a = \text{Move}(x, y) \\ \wedge \neg \exists z[\sigma(\text{On}(z, x))] \\ \wedge \neg \exists z[\sigma(\text{On}(z, y))] \\ \wedge x \neq y. \end{aligned}$$

Then P is equivalent to

$$\begin{aligned} \forall sxy.a = \text{Move}(x, y) \\ \wedge \neg \exists z[\text{State}[s](\text{On}(z, x))] \\ \wedge \neg \exists z[\text{State}[s](\text{On}(z, y))] \\ \wedge x \neq y \\ \supset \text{Possible}(a, s), \end{aligned}$$

or

$$\begin{aligned} \forall sxy.\neg \exists z[\text{State}[s](\text{On}(z, x))] \\ \wedge \neg \exists z[\text{State}[s](\text{On}(z, y))] \\ \wedge x \neq y \\ \supset \text{Possible}(\text{Move}(x, y), s). \end{aligned}$$

In the presence of (4), this can be rewritten as

$$\begin{aligned} \forall sxy.\neg \exists z \text{Holds}(\text{On}(z, x), s) \\ \wedge \neg \exists z \text{Holds}(\text{On}(z, y), s) \\ \wedge x \neq y \\ \supset \text{Possible}(\text{Move}(x, y), s), \end{aligned}$$

which is the universal closure of (11).

7. I ("the commonsense law of inertia") is the universal closure of (12), that is,

$$\begin{aligned} \forall fas.\text{Frame}(f) \\ \wedge \text{Possible}(a, s) \\ \wedge \neg \text{Noninertial}(f, a, s) \\ \supset [\text{Holds}(f, \text{Result}(a, s)) \equiv \text{Holds}(f, s)]. \end{aligned}$$

Finally, the circumscription policy of T is assumed to be (13).

3.2 STATEMENT OF THE MAIN THEOREM

The formulas (14)–(16), used in Section 2.5 for characterizing the extensions of the circumscribed predicates, make sense not only in the language of BW , but in any language L of the kind described in Section 3.1. The main theorem asserts that, under certain assumptions about $C_0(\sigma)$ and $R_0(a, f)$, the circumscriptions represented by the policy (13) have exactly the same effect as adding the formulas (14)–(16) to the axioms of T .

The following two conditions have to be imposed on $C_0(\sigma)$ and $R_0(a, f)$:

Condition A. The inessential axioms of T and the axiom F imply

$$R_0(a, f) \supset \text{Frame}(f). \quad (18)$$

Condition B. There exists a formula

$$\text{Compatible}(f_1, f_2),$$

containing no essential constants and no free variables other than f_1 and f_2 , such that the inessential axioms of T imply:

1. $\text{Compatible}(f_1, f_2) \equiv \text{Compatible}(f_2, f_1)$.
2. $C_0(\sigma) \equiv$
 $\forall f_1 f_2[\sigma(f_1) \wedge \sigma(f_2) \supset \text{Compatible}(f_1, f_2)]$.
3. $R_0(a, f_1) \wedge R_0(a, f_2) \supset \text{Compatible}(f_1, f_2)$.

Let us check that BW satisfies these conditions. For this theory, (18) is

$$\exists xy(a = \text{Move}(x, y) \wedge f = \text{On}(x, y)) \supset \text{Frame}(f),$$

which is a consequence of (4). We can take

$$\begin{aligned} \text{Compatible}(f_1, f_2) \\ \equiv \forall x[\text{About}(f_1, x) \wedge \text{About}(f_2, x) \supset f_1 = f_2] \end{aligned}$$

(About is defined in Section 2.5.) Part 1 of Condition B is obvious. To prove part 2, notice that its right-hand side can be written in the form

$$\begin{aligned} \forall f_1 f_2 x[\sigma(f_1) \wedge \sigma(f_2) \\ \wedge \text{About}(f_1, x) \wedge \text{About}(f_2, x) \\ \supset f_1 = f_2], \end{aligned}$$

or

$$\begin{aligned} \forall f_1 f_2 x[\sigma(f_1) \wedge \sigma(f_2) \\ \wedge f_1 = \text{Ontable}(x) \wedge f_2 = \text{Ontable}(x) \\ \supset f_1 = f_2] \\ \wedge \forall f_1 f_2 x y[\sigma(f_1) \wedge \sigma(f_2) \\ \wedge f_1 = \text{On}(x, y) \wedge f_2 = \text{Ontable}(x) \\ \supset f_1 = f_2] \\ \wedge \forall f_1 f_2 x y[\sigma(f_1) \wedge \sigma(f_2) \\ \wedge f_1 = \text{Ontable}(x) \wedge f_2 = \text{On}(x, y) \\ \supset f_1 = f_2] \\ \wedge \forall f_1 f_2 x y_1 y_2[\sigma(f_1) \wedge \sigma(f_2) \\ \wedge f_1 = \text{On}(x, y_1) \wedge f_2 = \text{On}(x, y_2) \\ \supset f_1 = f_2]. \end{aligned}$$

The first conjunctive term is trivial. The second is equivalent to

$$\begin{aligned} \forall xy[\sigma(\text{On}(x, y)) \wedge \sigma(\text{Ontable}(x)) \\ \supset \text{On}(x, y) = \text{Ontable}(x)]. \end{aligned}$$

In the presence of (3), this is equivalent to

$$\forall xy[\sigma(\text{On}(x, y)) \supset \neg \sigma(\text{Ontable}(x))],$$

which is one half of $C_0(\sigma)$. The third conjunctive term is equivalent to the second. The fourth term is equivalent to

$$\begin{aligned} \forall xy_1 y_2[\sigma(\text{On}(x, y_1)) \wedge \sigma(\text{On}(x, y_2)) \\ \supset \text{On}(x, y_1) = \text{On}(x, y_2)]. \end{aligned}$$

In the presence of (3), this is equivalent to

$$\forall xy_1y_2[\sigma(On(x, y_1)) \wedge \sigma(On(x, y_2)) \supset y_1 = y_2],$$

which gives the remaining half of $C_0(\sigma)$. Finally, part 3 of Condition B is

$$\begin{aligned} & \exists xy(a = Move(x, y) \\ & \quad \wedge f_1 = On(x, y)) \\ & \quad \wedge \exists xy(a = Move(x, y) \wedge f_2 = On(x, y)) \\ & \quad \supset \text{Compatible}(f_1, f_2). \end{aligned}$$

In the presence of (9), this is equivalent to the universal closure of

$$\begin{aligned} a = Move(x, y) \wedge f_1 = On(x, y) \wedge f_2 = On(x, y) \\ \supset \text{Compatible}(f_1, f_2)), \end{aligned}$$

which immediately follows from the definition of *Compatible*.

In the statement of the theorem, T is a circumspective theory of the kind described in Section 3.1.

Theorem. *If T satisfies Conditions A and B, then it is equivalent to the conjunction of its axioms and the formulas (14)–(16).*

3.3 DISCUSSION

We arrived at the class of theories described in Sections 3.1 and 3.2 by generalizing a single example—the theory of the blocks world from Section 2. As a result, all these theories share a number of “family traits,” inherited from their ancestor. The blocks world example has no initial situation or initial conditions; accordingly, the assumptions of the main theorem make it impossible to have initial conditions in the axiom set. (Initial conditions contain the predicate *Holds*, so that they cannot be included among the inessential axioms. On the other hand, they do not have any of the 7 forms that the essential axioms are allowed to have.) The immediate effect of $Move(x, y)$ is to make a certain fluent true, rather than false; accordingly, the main theorem assumes that the changes caused by all actions are “positive.” The blocks world example does not address the qualification problem; accordingly, the circumspective theories covered by the main theorem do not deal with it either. There are other limitations.

It seems, however, that it will not be difficult to prove analogs and extensions of the main theorem that overcome many of these limitations. We know, for instance, that the approach used in the blocks world example can handle initial conditions and temporal projection; it should be possible then to describe initial conditions in an abstract form, as one more kind of “essential axioms,” and prove the main theorem for systems with such axioms. It may be possible to prove similar theorems for some formalizations of action that

include continuous time and concurrency.⁶ Moreover, there can be many kinds of mathematical results confirming the adequacy of formalizations; results about the extents of the circumscribed predicates, as in the main theorem, represent only one of them. Assuming that the initial conditions provide the values of all frame fluents in the initial situation, we may be able to prove, for instance, that T decides every instance of the temporal projection problem. Hopefully, the main theorem can serve as a starting point for developing the theory of action along the lines of this metamathematical approach.

3.4 PAINTING BLOCKS

To illustrate the possibilities of the metamathematical approach, we will apply the main theorem to an extension of the blocks world example in which blocks can be not only moved, but also painted. We will see that, with small amount of additional work, we can use the main theorem to determine the effect of circumscription in the enhanced theory.

The theory *BW* is extended as follows. Variables for colors c, c_1, \dots are added to the language, along with the binary predicate *Color*, whose arguments are a block and a color, and the binary function *Paint*, whose arguments are a block and a color also, and whose value is an action. The new axioms are:

$$\begin{aligned} \text{Color}(x_1, c_1) = \text{Color}(x_2, c_2) \supset x_1 = x_2 \wedge c_1 = c_2, \\ \text{Color}(x, c) \neq \text{On}(y, z), \\ \text{Color}(x, c) \neq \text{Ontable}(y), \end{aligned} \tag{19}$$

$$\text{Frame}(\text{Color}(x, c)), \tag{20}$$

$$\begin{aligned} \text{Holds}(\text{Color}(x, c_1), s) \wedge \text{Holds}(\text{Color}(x, c_2), s) \\ \supset c_1 = c_2, \end{aligned} \tag{21}$$

$$\begin{aligned} \text{Paint}(x_1, c_1) = \text{Paint}(x_2, c_2) \supset x_1 = x_2 \wedge c_1 = c_2, \\ \text{Paint}(x, c) \neq \text{Move}(y, z), \end{aligned} \tag{22}$$

$$\begin{aligned} \text{Possible}(\text{Paint}(x, c), s) \\ \supset \text{Holds}(\text{Color}(x, c), \text{Result}(\text{Paint}(x, c), s)), \\ \text{Possible}(\text{Paint}(x, c), s). \end{aligned} \tag{23} \tag{24}$$

The extended theory satisfies the conditions of the main theorem, if the additional axioms are treated in the following way. Formulas (19) and (22) are included in the set of inessential axioms. The universal closure of (20) is appended to F as another conjunctive term. The axiom (21) is replaced by adding

$\forall c_1c_2[\sigma(\text{Color}(x, c_1)) \wedge \sigma(\text{Color}(x, c_2)) \supset c_1 = c_2]$ to $C_0(\sigma)$. Instead of including (23), we disjunctively append

$$\exists xc(a = \text{Paint}(x, c) \wedge f = \text{Color}(x, c))$$

⁶We argue in [Gelfond et al., 1991] that these ideas can be conveniently expressed in the language of the situation calculus, so that its attractive syntax should not be necessarily tied to the primitive ontology of action accepted in this paper.

to $R_0(a, f)$. Finally, instead of including (24), we disjunctively append

$$\exists xc(a = \text{Paint}(x, c))$$

to $P_0(a, \sigma)$.

The characterization of *Noninertial* given by the main theorem will allow us to prove, for instance, that moving blocks does not change their colors, and painting blocks does not change their locations.

4 PROOF OF THE MAIN THEOREM

The theorem follows from three lemmas, given in Section 4.3–4.5. Sections 4.1 and 4.2 contain some preliminary results.

4.1 A LEMMA ABOUT CIRCUMSCRIPTION

We will need the following general property of circumscription $\text{CIRC}[A(P, Z); P; Z]$, which generalizes Theorem 6.4 from [Genesereth and Nilsson, 1987].

Lemma 1. Let E be a predicate expression without parameters, containing neither P nor Z . If the sentences

$$A(P, Z) \supset \exists z A(E, z) \quad (25)$$

and

$$A(P, Z) \supset E \leq P \quad (26)$$

are universally valid, then so is the sentence

$$\text{CIRC}[A(P, Z); P; Z] \equiv A(P, Z) \wedge P = E. \quad (27)$$

Proof. To prove (27) left to right, assume $\text{CIRC}[A(P, Z); P; Z]$, that is,

$$A(P, Z) \wedge \neg \exists p z [A(p, z) \wedge p < P].$$

From the first conjunctive term we conclude, using (26), that $E \leq P$, and, using (25), that $\exists z A(E, z)$. From the second conjunctive term,

$$\neg \exists z [A(E, z) \wedge E < P].$$

The last two formulas imply $\neg(E < P)$. In combination with $E \leq P$, this gives $P = E$. Right to left: assume

$$A(P, Z) \wedge P = E. \quad (28)$$

Since (26) is universally valid, so is

$$\forall p z [A(p, z) \supset E \leq p] \quad (29)$$

(constants in a universally valid formula can be replaced by universally quantified variables). Assume $A(p, z) \wedge p < P$. Then, by (29), $E \leq p < P$, contrary to the second term of (28). This contradiction proves the second term of the circumscription in (27).

By essentially the same argument we can prove the relativized form of Lemma 1: For any set Γ of sentences containing neither P nor Z , if Γ implies (25) and (26) (that is, if these sentences are true in every model of Γ), then Γ implies (27).

4.2 THE FUNCTION ρ

In the remaining part of Section 4, T is a theory satisfying Conditions A and B. By A we denote the conjunction of all axioms of T , and by A_0 the conjunction of the inessential axioms, so that

$$A \equiv A_0 \wedge F \wedge S \wedge U \wedge C \wedge R \wedge P \wedge I.$$

Define the function ρ by:

$$\begin{aligned} \rho[a, \sigma] &= \lambda f. [\sigma(f) \vee R_0(a, f)] \\ &\wedge \forall f_1 [R_0(a, f_1) \supset \text{Compatible}(f, f_1)]. \end{aligned}$$

Intuitively, this is the counterpart of the function *Result* which works on states instead of situations.

The following two lemmas summarize the properties of ρ used in the proof of the theorem.

Lemma 2. A_0 and F imply

$$\text{IsState}(\sigma) \supset \text{IsState}(\rho[a, \sigma]), \quad (30)$$

$$C_0(\sigma) \supset C_0(\rho[a, \sigma]), \quad (31)$$

and

$$R_0(a, f) \supset \rho[a, \sigma](f). \quad (32)$$

Proof. The fact that A_0 and F imply (30) follows from Condition A. To prove (31), assume A_0 and $C_0(\sigma)$. Then, according to part 2 of Condition B,

$$\forall f_1 f_2 [\sigma(f_1) \wedge \sigma(f_2) \supset \text{Compatible}(f_1, f_2)]. \quad (33)$$

We need to prove $C_0(\rho[a, \sigma])$, that is,

$$\forall f_1 f_2 [\rho[a, \sigma](f_1) \wedge \rho[a, \sigma](f_2) \supset \text{Compatible}(f_1, f_2)].$$

Assume $\rho[a, \sigma](f_1)$ and $\rho[a, \sigma](f_2)$. Then, by the definition of ρ ,

$$\sigma(f_1) \vee R_0(a, f_1), \quad (34)$$

$$R_0(a, f_2) \supset \text{Compatible}(f_1, f_2), \quad (35)$$

$$\sigma(f_2) \vee R_0(a, f_2), \quad (36)$$

$$R_0(a, f_1) \supset \text{Compatible}(f_2, f_1). \quad (37)$$

Our goal is to derive $\text{Compatible}(f_1, f_2)$. If $\sigma(f_1)$ and $\sigma(f_2)$, then this conclusion follows by (33). If not, then, by (34) and (36), $R_0(a, f_1)$ or $R_0(a, f_2)$. In the first case, use (37) and part 1 of Condition B; in the second case, use (35). Formula (31) is proved. To prove (32), assume $R_0(a, f)$. According to the definition of ρ , we need only to check that $R_0(a, f_1) \supset \text{Compatible}(f, f_1)$; this follows from part 3 of Condition B.

Lemma 3. A_0 , F and C imply

$$\text{IsState}(\rho[a, \text{State}[s]]) \wedge C_0(\rho[a, \text{State}[s]]). \quad (38)$$

Proof: (38) follows from (30), (31) and C .

4.3 THE EFFECT OF CIRCUMSCRIBING Frame

Lemma 4. *The circumscription*

$\text{CIRC}[A; \text{Frame}; \text{Inconsistent}, \text{Noninertial}] \quad (39)$
is equivalent to the conjunction of A and (14).

Proof. By Proposition 2 from [Lifschitz, 1985],

$$\begin{aligned} &\text{CIRC}[A; \text{Frame}; \text{Inconsistent}, \text{Noninertial}] \\ &\equiv A \wedge \text{CIRC}[(\exists \text{Inconsistent}, \text{Noninertial}.A); \\ &\quad \text{Frame}] \\ &\equiv A \wedge \text{CIRC}[A_0 \wedge F \wedge (\exists \text{Inconsistent}.S) \\ &\quad \wedge U \wedge C \wedge R \wedge P \wedge (\exists \text{Noninertial}.I); \text{Frame}] \\ &\equiv A \wedge \text{CIRC}[A_0 \wedge F \wedge U \wedge C \wedge R \wedge P; \text{Frame}]. \end{aligned}$$

The formulas A_0 , R and P do not contain *Frame*. Both occurrences of *Frame* in U (as parts of $\text{IsState}[\sigma_1]$ and $\text{IsState}[\sigma_2]$) are negative. According to part 2 of Condition B, C is equivalent, in the presence of A_0 , to a formula not containing *Frame*. Consequently, the conjunction $A_0 \wedge U \wedge C \wedge R \wedge P$ is equivalent to a formula negative relative to *Frame*. By the lemma from Section 4 of [Lifschitz, 1987], it follows that

$$\begin{aligned} &\text{CIRC}[A_0 \wedge F \wedge U \wedge C \wedge R \wedge P; \text{Frame}] \\ &\equiv \text{CIRC}[F; \text{Frame}] \wedge A_0 \wedge U \wedge R \wedge C \wedge P. \end{aligned}$$

Hence

$$\begin{aligned} &\text{CIRC}[A; \text{Frame}; \text{Inconsistent}, \text{Noninertial}] \\ &\equiv A \wedge \text{CIRC}[F; \text{Frame}]. \end{aligned}$$

4.4 THE EFFECT OF CIRCUMSCRIBING Inconsistent

Lemma 5. *The circumscription*

$$\text{CIRC}[A; \text{Inconsistent}, \text{Holds}, \text{Result}, \text{Possible}, \text{Noninertial}] \quad (40)$$

is equivalent to the conjunction of A and (16).

Proof. Notice first that (40) is equivalent to

$$A \wedge \text{CIRC}[(\exists \text{Possible}, \text{Noninertial}.A; \text{Inconsistent}; \text{Holds}, \text{Result})]$$

(apply Proposition 2 from [Lifschitz, 1985] to both circumscriptions). The formula

$$\exists \text{Possible}, \text{Noninertial}.A$$

can be simplified as follows:

$$\begin{aligned} &\exists \text{Possible}, \text{Noninertial}.A \\ &\equiv A_0 \wedge F \wedge S \wedge U \wedge C \wedge (\exists \text{Possible}.R \wedge P) \\ &\quad \wedge (\exists \text{Noninertial}.I) \\ &\equiv A_0 \wedge F \wedge S \wedge U \wedge C \wedge (\exists \text{Possible}.R \wedge P) \\ &\equiv A_0 \wedge F \wedge S \wedge U \wedge C \wedge R^*, \end{aligned}$$

where R^* stands for

$$P_0(a, \text{State}[s]) \wedge R_0(a, f) \supset \text{Holds}(f, \text{Result}(a, s)).$$

It follows that (40) is equivalent to

$$A \wedge \text{CIRC}[A_0 \wedge F \wedge S \wedge U \wedge C \wedge R^*; \text{Inconsistent}; \text{Holds}, \text{Result}].$$

We will compute this circumscription using Lemma 1, with the expression

$$\lambda\sigma[\text{IsState}(\sigma) \wedge \neg C_0(\sigma)]$$

as E . We need to prove

$$\begin{aligned} &A_0 \wedge F \wedge S(\text{Inconsistent}, \text{Holds}) \wedge U \\ &\wedge C(\text{Holds}) \wedge R^*(\text{Holds}, \text{Result}) \\ &\supset \exists \text{holds}, \text{result}[A_0 \wedge F \wedge S(E, \text{holds}) \\ &\quad \wedge U \wedge C(\text{holds}) \wedge R^*(\text{holds}, \text{result})] \end{aligned}$$

and

$$\begin{aligned} &A_0 \wedge F \wedge S(\text{Inconsistent}, \text{Holds}) \wedge U \\ &\wedge C(\text{Holds}) \wedge R^*(\text{Holds}, \text{Result}) \\ &\supset E \leq \text{Inconsistent}. \end{aligned}$$

In other words, assuming

$$A_0 \wedge F \wedge S(\text{Inconsistent}, \text{Holds}) \wedge U \\ \wedge C(\text{Holds}) \wedge R^*(\text{Holds}, \text{Result}),$$

we need to select *holds* and *result*, and prove:

$$\begin{aligned} &S(E, \text{holds}), \\ &C(\text{holds}), \\ &R^*(\text{holds}, \text{result}), \\ &E \leq \text{Inconsistent}. \end{aligned}$$

The predicate *holds* is defined by cases, depending on whether or not there is a σ satisfying the condition

$$\text{IsState}(\sigma) \wedge C_0(\sigma) \wedge s = \text{Sit}(\sigma) \quad (41)$$

$$\text{holds}(f, s) \equiv \begin{cases} \sigma(f), & \text{if } \sigma \text{ satisfies (41),} \\ \text{false}, & \text{if there is no such } \sigma. \end{cases}$$

The correctness of this definition follows from the fact that, by U , there can be at most one σ satisfying (41). The function *result* is defined by

$$\text{result}(a, s) = \text{Sit}(\rho[a, \text{State}[s]]).$$

The formula $S(E, \text{holds})$, that is,

$$\begin{aligned} &\text{IsState}(\sigma) \wedge \neg[\text{IsState}(\sigma) \wedge \neg C_0(\sigma)] \\ &\supset \sigma = \lambda f[\text{Frame}(f) \wedge \text{holds}(f, \text{Sit}(\sigma))], \end{aligned}$$

is propositionally equivalent to

$$\begin{aligned} &\text{IsState}(\sigma) \wedge C_0(\sigma) \\ &\supset \sigma = \lambda f[\text{Frame}(f) \wedge \text{holds}(f, \text{Sit}(\sigma))]. \end{aligned}$$

Assume $\text{IsState}(\sigma)$ and $C_0(\sigma)$. Then, using the definition of *holds*, we compute:

$$\begin{aligned} &\lambda f[\text{Frame}(f) \wedge \text{holds}(f, \text{Sit}(\sigma))] \\ &= \lambda f[\text{Frame}(f) \wedge \sigma(f)] \\ &= \lambda f.\sigma(f) = \sigma. \end{aligned}$$

The formula $C(\text{holds})$ can be written as

$$C_0(\lambda f[\text{Frame}(f) \wedge \text{holds}(f, s)]).$$

If σ is such that $IsState(\sigma) \wedge C_0(\sigma) \wedge s = Sit(\sigma)$, then, according to the definition of *holds*, $holds(f, s) \equiv \sigma(f)$, so that $C(holds)$ is equivalent to

$$C_0(\lambda f[Frame(f) \wedge \sigma(f)]).$$

Since $IsState(\sigma)$, this is the same as $C_0(\sigma)$. On the other hand, if there is no σ satisfying the condition $IsState(\sigma) \wedge C_0(\sigma) \wedge s = Sit(\sigma)$, then $C(holds)$ becomes $C_0(\lambda f.\text{false})$, which follows from part 2 of Condition B.

The formula $R^*(holds, result)$ can be written as

$$\begin{aligned} P_0(a, \lambda f(Frame(f) \wedge holds(f, s))) \\ \wedge R_0(a, f) \supset holds(f, result(a, s)). \end{aligned}$$

We can actually prove the stronger formula

$$R_0(a, f) \supset holds(f, result(a, s)),$$

that is,

$$R_0(a, f) \supset holds(f, Sit(\rho[a, State[s]])).$$

Using (38) and the definition of *holds*, this can be further rewritten as

$$R_0(a, f) \supset \rho[a, State[s]](f),$$

which follows from (32).

Finally, $E \leq Inconsistent$ stands for

$$IsState(\sigma) \wedge \neg C_0(\sigma) \supset Inconsistent(\sigma).$$

Assume $IsState(\sigma)$ and $\neg Inconsistent(\sigma)$. It follows then from S that $\sigma = State[s]$ for some s . Then $C_0(State[s])$ follows from C .

4.5 THE EFFECT OF CIRCUMSCRIBING Noninertial

Lemma 6. Assuming (15), the circumscription

$$CIRC[A; Noninertial, Result] \quad (42)$$

is equivalent to the conjunction of A and (16).

Proof. We will use the relativized form of Lemma 1, with (15) as Γ and the expression

$$\begin{aligned} \lambda f as[Frame(f) \wedge Possible(a, s) \\ \wedge Affected(f, a, State[s])] \end{aligned}$$

as E . We need to prove that (15) implies

$$A(Noninertial, Result) \supset \exists result A(E, result)$$

and

$$A(Noninertial, Result) \supset E \leq Noninertial.$$

Assume $A(Noninertial, Result)$, and define *result* as in the proof of Lemma 5. Since the only parts of A that contain *Noninertial* or *Result* are R and I , we need only to prove:

$$\begin{aligned} & R(result), \\ & I(E, Result), \\ & E \leq Noninertial. \end{aligned}$$

The following relationship between *result* and ρ will be used in the proof:

$$State[result(a, s)] = \rho[a, State[s]]. \quad (43)$$

To prove this, notice that (38) and (15) imply $\neg Inconsistent(\rho[a, State[s]])$. Using (38) and S , we conclude that the right-hand side of (43) equals $State[Sit(\rho[a, State[s]])]$. By the definition of *result*, this is the same as the right-hand side of (43).

The formula $R(result)$ is

$$Possible(a, s) \wedge R_0(a, f) \supset Holds(f, result(a, s)).$$

We will prove a stronger formula:

$$R_0(a, f) \supset Holds(f, result(a, s)).$$

Assume $R_0(a, f)$. Then, by (32), $\rho[a, State[s]](f)$. It follows by (43) that $State[result(a, s)](f)$, and consequently $Holds(f, result(a, s))$.

The formula $I(E, holds)$ is

$$\begin{aligned} Frame(f) \wedge Possible(a, s) \wedge \neg E(f, a, s) \\ \supset [Holds(f, result(a, s)) \equiv Holds(f, s)], \end{aligned}$$

which is propositionally equivalent to

$$\begin{aligned} Frame(f) \\ \wedge Possible(a, s) \\ \wedge \neg Affected(f, a, State[s]) \\ \supset [Holds(f, result(a, s)) \equiv Holds(f, s)]. \end{aligned}$$

Assume that $Frame(f)$ and $Possible(a, s)$, but

$$\neg[Holds(f, result(a, s)) \equiv Holds(f, s)]. \quad (44)$$

Using (43), we can rewrite this in the form

$$\neg\{\rho[a, State[s]](f) \equiv State[s](f)\},$$

which, by the definition of ρ , is the same as

$$\begin{aligned} \neg\{\{(State[s](f) \vee R_0(a, f)) \\ \wedge \forall f_1[R_0(a, f_1) \supset Compatible(f, f_1)]\} \\ \equiv State[s](f)\}. \end{aligned} \quad (45)$$

Our goal is to prove

$$Affected(f, a, State[s]),$$

that is,

$$\begin{aligned} IsState(\sigma_1) \wedge C_0(\sigma_1) \wedge [\sigma_1(f) \equiv State[s](f)] \\ \supset \exists f_1[R_0(a, f_1) \wedge \neg \sigma_1(f_1)]. \end{aligned} \quad (46)$$

Assume $IsState(\sigma_1)$, $C_0(\sigma_1)$ and

$$\sigma_1(f) \equiv State[s](f).$$

Using the last equivalence $\sigma_1(f) \equiv State[s](f)$, we can rewrite (45) in the form

$$\begin{aligned} \neg\{\{(\sigma_1(f) \vee R_0(a, f)) \\ \wedge \forall f_1[R_0(a, f_1) \supset Compatible(f, f_1)]\} \equiv \sigma_1(f)\}. \end{aligned} \quad (47)$$

Case 1: $R_0(a, f)$. Then, by part 3 of Condition B,

$$\forall f_1[R_0(a, f_1) \supset \text{Compatible}(f, f_1)],$$

so that the left-hand side of the equivalence in (47) is true, and (47) implies $\neg\sigma_1(f)$. We see that the consequent of (46) is true for $f_1 = f$. Case 2: $\neg R_0(a, f)$. Then (47) can be rewritten as

$$\begin{aligned} \neg\{ & \{(\sigma_1(f) \wedge \forall f_1[R_0(a, f_1) \supset \text{Compatible}(f, f_1)]) \\ & \equiv \sigma_1(f)\} \end{aligned}$$

This is only possible when $\sigma_1(f)$ and

$$\neg\forall f_1[R_0(a, f_1) \supset \text{Compatible}(f, f_1)].$$

Take f_1 such that $R_0(a, f_1)$ and $\neg\text{Compatible}(f, f_1)$. By part 2 of Condition B,

$$\sigma_1(f) \wedge \sigma_1(f_1) \supset \text{Compatible}(f, f_1).$$

Consequently, $\neg\sigma_1(f_1)$, which again proves the consequent of (46).

It remains to show that $E \leq \text{Noninertial}$, that is,

$$\begin{aligned} \text{Frame}(f) \wedge \text{Possible}(a, s) \wedge \text{Affected}(f, a, \text{State}[s]) \\ \supset \text{Noninertial}(f, a, s). \end{aligned}$$

We will assume $\text{Frame}(f)$, $\text{Possible}(a, s)$ and

$$\neg\text{Noninertial}(f, a, s),$$

and prove

$$\neg\text{Affected}(f, a, \text{State}[s]),$$

that is,

$$\begin{aligned} \exists\sigma_1[\text{IsState}(\sigma_1) \wedge C_0(\sigma_1) \wedge (\sigma_1(f) \equiv \text{State}[s](f) \\ \wedge \neg\exists f_1[R_0(a, f_1) \wedge \neg\sigma_1(f_1)]). \end{aligned}$$

By I,

$$\text{Holds}(f, \text{Result}(a, s)) \equiv \text{Holds}(f, s). \quad (48)$$

Let us check that $\sigma_1 = \text{State}[\text{Result}(a, s)]$ has all required properties. Clearly, $\text{IsState}(\sigma_1)$; C implies $C_0(\sigma_1)$; (48) implies $\sigma_1(f) \equiv \text{State}[s](f)$. Let f_1 be such that $R_0(a, f_1)$. Then, by R, $\text{Holds}(f_1, \text{Result}(a, s))$, that is, $\sigma_1(f_1)$.

Acknowledgments

The author would like to thank Michael Gelfond, Fangzhen Lin, John McCarthy and Raymond Reiter for useful discussions on the subject of this paper and for comments on an earlier draft. He is also grateful to the anonymous referees for their comments on the abstract of this paper. The work on the paper began when the author worked at Stanford University and was supported in part by NSF grant IRI-89-04611 and by DARPA under Contract N00039-84-C-0211.

References

- [Baker and Ginsberg, 1989] Andrew Baker and Matthew Ginsberg. Temporal projection and explanation. In *Proc. of IJCAI-89*, pages 906–911, 1989.

[Baker, 1989] Andrew Baker. A simple solution to the Yale Shooting Problem. In Ronald Brachman, Hector Levesque, and Raymond Reiter, editors, *Proc. of the First Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 11–20, 1989.

[Gelfond et al., 1991] Michael Gelfond, Vladimir Lifschitz, and Arkady Rabinov. What are the limitations of the situation calculus? In Robert Boyer, editor, *Essays for Bledsoe*. Kluwer Academic, Dordrecht, 1991. To appear.

[Genesereth and Nilsson, 1987]

Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, San Mateo, CA, 1987.

[Lifschitz, 1985] Vladimir Lifschitz. Computing circumscription. In *Proc. of IJCAI-85*, pages 121–127, 1985.

[Lifschitz, 1987] Vladimir Lifschitz. Pointwise circumscription. In Matthew Ginsberg, editor, *Readings in nonmonotonic reasoning*, pages 179–193. Morgan Kaufmann, San Mateo, CA, 1987.

[Lifschitz, 1990] Vladimir Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[McCarthy and Hayes, 1969]

John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[McCarthy, 1986] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 26(3):89–116, 1986.

On the Difference between Updating a Knowledge Base and Revising it

Hirofumi Katsuno

NTT Basic Research Laboratories
Musashino-shi, Tokyo 180, Japan
`katsuno@ntt-20.ntt.jp`

Alberto O. Mendelzon

Computer Systems Research Institute
University of Toronto
Toronto, Canada M5S 1A4
`mendel@db.toronto.edu`

Abstract

We explore a fundamental distinction between two kinds of modifications to a knowledge base represented by a propositional theory. The first one, *update*, consists of bringing the knowledge base up to date when the world described by it changes. The second type of modification, *revision*, is used when we are obtaining new information about a static world. Alchourrón, Gärdenfors and Makinson have proposed and studied a set of rationality postulates that they argue must be satisfied by every operator for theory change. We show these postulates apply only to revision, not to update. We give a new set of postulates that apply to update operators, and characterize all operators that satisfy the postulates in terms of a set of partial orders defined among possible worlds.

A dual operation of revision is contraction, the act of giving up a belief. We propose a new operation called *erasure*. Erasure is to contraction as update is to revision. We show that Winslett's *Forget* operator is a special case of a general operation called *symmetric erasure*, defined in terms of erasure.

1 INTRODUCTION

Consider a knowledge base represented by a theory ψ of some logic, say propositional logic. We want to incorporate into ψ a new fact, represented by a sentence μ of the same language. What should the resulting theory be? A growing body of work [3, 9, 15, 17] takes as a departure point the *rationality postulates* proposed by Alchourrón, Gärdenfors and Makinson [1]. These are rules that every adequate revision operator should be expected to satisfy. For example: the new fact μ must be a consequence of the revised knowledge base.

In this paper, we argue that no such set of postulates will be adequate for every application. In particular,

we make a fundamental distinction between two kinds of modifications to a knowledge base. The first one, *update*, consists of bringing the knowledge base up to date when the world described by it changes. For example, most database updates are of this variety, e.g. "increase Joe's salary by 5%". Another example is the incorporation into the knowledge base of changes caused in the world by the actions of a robot [7, 19, 21]. We show that the AGM postulates must be drastically modified to describe update.

The second type of modification, *revision*, is used when we are obtaining new information about a static world. For example, we may be trying to diagnose a faulty circuit and want to incorporate into the knowledge base the results of successive tests, where newer results may contradict old ones. We claim the AGM postulates describe only revision.

The distinction between update and revision was made by Keller and Winslett [11] in the context of extended relational databases. They distinguished *change-recording updates* (which we call updates) and *knowledge-adding updates* (which we call revisions). Our work extends theirs in several ways. We formalize the distinction, which they made informally. We provide an axiomatization for update obtained from the AGM. Keller and Winslett's work does not treat inconsistent knowledge-bases or addition of inconsistent knowledge, while ours does. And we treat arbitrary propositional knowledge-bases, while their setting is relational databases extended with null values and disjunction.

Rao and Foo [17] extend the AGM postulates in order to apply them to reasoning about action. They introduce the notion of time and consider a modal logic. However, they do not identify the difference between revision and update. In this paper we clarify exactly why the postulates apply to revision but not to update. We give a new set of postulates that apply to update operators, and characterize the set of operators that satisfy the postulates in terms of a set of partial orders defined among possible worlds.

The difference between the postulates for revision and for update can be explained intuitively as follows. Suppose knowledge base ψ is to be revised with sentence μ . Revision methods that satisfy the AGM postulates are exactly those that select from the models of μ those that are "closest" to models of ψ , where the notion of closeness is defined by an ordering relationship among models that satisfies certain conditions [10]. The models selected determine the new theory, which we denote by $\psi \circ \mu$. On the other hand, update methods select, for each model M of the knowledge base ψ , the set of models of μ that are closest to M . The new theory describes the union of all such models. Suppose that ψ has exactly two models, I and J ; that is, there are two possible worlds described by the knowledge base. Suppose that μ describes exactly two worlds, K and L , and that K is "closer" to I than L is, and K is also closer to I than L is to J . Then K is selected for the new knowledge base, but L is not. Note the knowledge base has effectively forgotten that J used to be a possible world; the new fact μ has been used as evidence for the retroactive impossibility of J . That is, not only do we refuse to have J as a model of the new knowledge base, but we also conclude that J should not have been in the old knowledge base to begin with.

If we are doing revisions, this behaviour is rational. Since the real world has not changed, and μ has to be true in all the new possible worlds, we can forget about some of the old possible worlds on the grounds that they are too different from what we now know to be the case. On the other hand, suppose we are doing updates. The models of ψ are possible worlds; we think one of them is the real world, but we do not know which one. Now the real world has changed; we examine each of the old possible worlds and find the minimal way of changing each one of them so that it becomes a model of μ . The fact that the real world has changed gives us no grounds to conclude that some of the old worlds were actually not possible.

To illustrate this distinction between update and revision, let us consider two examples which are formally identical to the one above but have different intuitively desirable results. First, in the spirit of Ginsberg and Smith [8] and Winslett [20], suppose our knowledge base describes five objects A,B,C,D,E inside a room. There is a table in the room, and objects may be on or off the table. The sentence a means "object A is on the table," and similarly for sentences b,c,d , and e . The knowledge base ψ is the sentence

$$(a \wedge \neg b \wedge \neg c \wedge \neg d \wedge \neg e) \vee (\neg a \wedge b \wedge c \wedge d \wedge e).$$

That is, either object A is on the table by itself, or objects C,D and E are. This knowledge base has exactly two models I and J . We send a robot into the room, instructing it to achieve a situation in which all or none of the objects are on the table. This change can be modelled by incorporating the following sen-

tence μ :

$$(a \wedge b \wedge c \wedge d \wedge e) \vee (\neg a \wedge \neg b \wedge \neg c \wedge \neg d \wedge \neg e).$$

Let us take Dalal's notion of "closeness" and the revision operator that results [4]. According to this measure, the distance between two models is simply the number of propositional letters on which they differ. The models selected for the new KB will be those models of μ which are at minimal distance from models of ψ . Now K , the model where nothing is on the table, is at distance 1 from I (the model where A is on the table) and at distance 3 from J (the model where C,D and E are). On the other hand L , the model where every object is on the table, is at distance 4 from I and 2 from J . Dalal's revision operator will therefore select K as the only model of the new knowledge base. But intuitively, it seems clear that this is incorrect. After the robot is done, all we know is that either all objects are on the table or all are off; there is no reason to conclude that they are all off, which is what revision does.

Consider now an example that is formally identical, but where the desired result is given by revision, not by update.

Suppose the knowledge base describes the state of a five bit register which we read through noisy communication lines. Each of the propositional letters a,b,c,d,e now represents one bit. The state of the register is unchanging. Two different readings have been obtained: 10000 and 00111. By an independent analysis of the circuits that control the register, we learn that all bits must have the same value. That is, only 11111 and 00000 are possible patterns. Dalal's revision method tells us to keep 00000 as the new knowledge base; that is, we conclude that 00111 is relatively too far from the possible patterns to be an acceptable result. It might be argued that it is better to forget the two readings in the KB and just keep both 00000 and 11111 as possible worlds. However, consider an example in which the register is thousands of bits long, the two readings agree on every bit except the first five, and the new fact only says that the first five bits must be all 0's or all 1's. It is clearly a waste of information now to discard the old KB and just keep the new fact.

With this motivation, let us postulate that an update method should give each of the old possible worlds equal consideration. One way of capturing this condition syntactically is to require that the result of updating $\psi \vee \phi$ with μ be equivalent to the disjunction of ψ updated with μ and ϕ updated with μ . Let us call this the *disjunction rule*. This rule turns out to have far-reaching consequences. In particular, consider the case where μ is consistent with ψ , that is, no conflict exists. The AGM postulates require the result of the revision to be simply the conjunction of ψ and μ . As we will see, this apparently obvious requirement is inconsistent with the disjunction rule.

The outline of the paper is as follows. In Section 2 we give preliminaries. We review the AGM postulates and our characterization from [10] of all revision methods that satisfy the postulates in terms of a pre-order among models. In Section 3 we define the update operation and give a set of rationality postulates for it. We show that these postulates characterize all update operators that select for each model I of ψ those models of μ that are “closest” to I in a certain sense. In Section 4, we discuss briefly how update and revision could be combined for reasoning about action. In Section 5, we propose a new operation called *erasure*. Erasure is the analogue of contraction [1, 14] for update operators. We show that Winslett’s *Forget* operator is a special case of *symmetric erasure*, an operator defined in terms of erasure.

2 PRELIMINARIES

Throughout this paper, we consider a finitary propositional language L , and we denote the set consisting of all the propositional letters in L by Ξ . We represent a knowledge base by a propositional formula ψ . An *interpretation* of L is a function from Ξ to $\{\text{T}, \text{F}\}$. A *model* of a propositional formula ψ is an interpretation that makes ψ true in the usual sense. $\text{Mod}(\psi)$ denotes the set of all the models of ψ . The knowledge base ψ may be *inconsistent*, in which case $\text{Mod}(\psi) = \emptyset$. A propositional formula ϕ is *complete* if for any propositional formula, μ , ϕ implies μ or ϕ implies $\neg\mu$.

2.1 REVISION AND THE AGM POSTULATES

Given a knowledge base ψ and a sentence μ , $\psi \circ \mu$ denotes the *revision* of ψ by μ ; that is, the new knowledge base obtained by adding new knowledge μ to the old knowledge base ψ .

Alchourrón, Gärdenfors and Makinson propose eight postulates, $(G^*1)\sim(G^*8)$, which they argue must be satisfied by any reasonable revision function. By specializing to the case of propositional logic and rephrasing them in terms of finite covers for infinite “knowledge sets,” the postulates become the six rules below. See [10, 11] for a discussion of the intuitive meaning and formal properties of these rules.

- (R1) $\psi \circ \mu$ implies μ .
- (R2) If $\psi \wedge \mu$ is satisfiable then $\psi \circ \mu \equiv \psi \wedge \mu$.
- (R3) If μ is satisfiable then $\psi \circ \mu$ is also satisfiable.
- (R4) If $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$ then $\psi_1 \circ \mu_1 \equiv \psi_2 \circ \mu_2$.
- (R5) $(\psi \circ \mu) \wedge \phi$ implies $\psi \circ (\mu \wedge \phi)$.
- (R6) If $(\psi \circ \mu) \wedge \phi$ is satisfiable then $\psi \circ (\mu \wedge \phi)$ implies $(\psi \circ \mu) \wedge \phi$.

2.2 ORDERS BETWEEN INTERPRETATIONS

The postulates (R5) and (R6) represent the condition that revision be accomplished with minimal change. In [10], we gave a model theoretic characterization of minimal change.

Let \mathcal{I} be the set of all the interpretations of L . A *pre-order* \leq over \mathcal{I} is a reflexive and transitive relation on \mathcal{I} . We define $<$ as $I < I'$ if and only if $I \leq I'$ and $I' \not\leq I$. A pre-order is *total* if for every $I, J \in \mathcal{I}$, either $I \leq J$ or $J \leq I$. Consider a function that assigns to each propositional formula ψ a pre-order \leq_ψ over \mathcal{I} . We say this assignment is *faithful*¹ if the following three conditions hold:

1. If $I, I' \in \text{Mod}(\psi)$ then $I <_\psi I'$ does not hold.
2. If $I \in \text{Mod}(\psi)$ and $I' \notin \text{Mod}(\psi)$ then $I <_\psi I'$ holds.
3. If $\psi \equiv \phi$, then $\leq_\psi = \leq_\phi$.

That is, a model of ψ cannot be strictly less than any other model of ψ and must be strictly less than any non-model of ψ .

Let \mathcal{M} be a subset of \mathcal{I} . An interpretation I is minimal in \mathcal{M} with respect to \leq_ψ if $I \in \mathcal{M}$ and there is no $I' \in \mathcal{M}$ such that $I' <_\psi I$. Let $\text{Min}(\mathcal{M}, \leq_\psi)$ be the set of all $I \in \mathcal{M}$ such that I is minimal in \mathcal{M} with respect to \leq_ψ . The following characterization of all revision operators that satisfy the postulates was established in [10].

Theorem 2.1 *Revision operator \circ satisfies Conditions (R1)~(R6) if and only if there exists a faithful assignment that maps each KB ψ to a total pre-order \leq_ψ such that $\text{Mod}(\psi \circ \mu) = \text{Min}(\text{Mod}(\mu), \leq_\psi)$.*

3 UPDATE

In this section we axiomatize all update operators that can be defined by partial orders or partial pre-orders over interpretations. The class of operators defined generalizes Winslett’s *Possible Models Approach* (PMA) [20, 21]. Winslett argues that the PMA is suitable for reasoning about action in certain applications. According to our classification, the PMA is an update operator, because it changes each possible world independently. For background, we review this approach first.

3.1 POSSIBLE MODELS APPROACH

Let ψ be a KB and μ a new sentence. We denote the PMA operator by \diamond_{pma} . For each model I of ψ ,

¹The term *persistent* was used instead of “faithful” in [10].

the PMA selects from the models of μ those which are “closest” to I . The models of the new KB $(\psi \diamond_{pma} \mu)$ are the union of these selected models. Formally, the PMA is defined by

$$\text{Mod}(\psi \diamond_{pma} \mu) = \bigcup_{I \in \text{Mod}(\psi)} \text{Incorporate}(\text{Mod}(\mu), I),$$

where $\text{Incorporate}(\text{Mod}(\mu), I)$ is the set of models that are “closest” to I in $\text{Mod}(\mu)$.

The closeness between two interpretations, I and J is measured by the set $\text{Diff}(I, J)$ of propositional letters that have different truth values under I and J . For two interpretations, J_1 and J_2 , J_1 is closer to I than J_2 (denoted by $J_1 \leq_{I,pma} J_2$) if and only if $\text{Diff}(I, J_1)$ is a subset of $\text{Diff}(I, J_2)$. Then, $\text{Incorporate}(\text{Mod}(\mu), I)$ is the set of all the minimal elements with respect to $\leq_{I,pma}$ in the set $\text{Mod}(\mu)$, that is, $\text{Min}(\text{Mod}(\mu), \leq_{I,pma})$.

Example 3.1 Let L have only two propositional letters, b and m . Let $\psi \equiv (b \wedge \neg m) \vee (\neg b \wedge m)$ and $\mu \equiv b$. Then, $I = \langle F, T \rangle$ is a model of ψ . $J_1 = \langle T, T \rangle$ and $J_2 = \langle T, F \rangle$ are two models of μ . $J_1 \leq_{I,pma} J_2$ follows from the fact $\text{Diff}(I, J_1) = \{b\}$ is a subset of $\text{Diff}(I, J_2) = \{b, m\}$. Similarly, by considering the case where J_2 is a model of ψ , we obtain $\psi \diamond_{pma} \mu \equiv b$.

To interpret this example in the context of [20, 21], let us go back to a room with two objects in it, a book and a magazine. Suppose b means the book is on the floor, and m means the magazine is on the floor. Then, ψ states that either the book is on the floor but not both. Now, we order a robot to put the book on the floor. The result of this action should be represented by the revision of ψ with b . After the robot puts the book on the floor, all we know is b , and this is in fact the result of applying the PMA. Note that ψ is consistent with μ . According to revision postulate (R2), the result of $\psi \circ \mu$ should therefore be $\psi \wedge \mu$, that is, $b \wedge \neg m$. But why should we conclude that the magazine is not on the floor?

3.2 POSTULATES FOR UPDATE

The PMA is defined in terms of a certain partial order over interpretations. This subsection generalizes the PMA by axiomatizing all update operators that can be defined by partial orders or partial pre-orders over interpretations.

We use $\psi \diamond \mu$ to denote the result of updating KB ψ with sentence μ .

Our postulates for update are:

- (U1) $\psi \diamond \mu$ implies μ .
- (U2) If ψ implies μ then $\psi \diamond \mu$ is equivalent to ψ .
- (U3) If both ψ and μ are satisfiable then $\psi \diamond \mu$ is also satisfiable.

- (U4) If $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$ then $\psi_1 \diamond \mu_1 \equiv \psi_2 \diamond \mu_2$.
- (U5) $(\psi \diamond \mu) \wedge \phi$ implies $\psi \diamond (\mu \wedge \phi)$.
- (U6) If $\psi \diamond \mu_1$ implies μ_2 and $\psi \diamond \mu_2$ implies μ_1 then $\psi \diamond \mu_1 \equiv \psi \diamond \mu_2$.
- (U7) If ψ is complete then $(\psi \diamond \mu_1) \wedge (\psi \diamond \mu_2)$ implies $\psi \diamond (\mu_1 \vee \mu_2)$.
- (U8) $(\psi_1 \vee \psi_2) \diamond \mu \equiv (\psi_1 \diamond \mu) \vee (\psi_2 \diamond \mu)$.

Postulates (U1)~(U5) correspond directly to the corresponding postulates for revision given in Section 2. Note that postulate (U2) says that if a new sentence μ is derivable from KB ψ , then updating by μ does not influence the KB. In the case where ψ is consistent, postulate (U2) is strictly weaker than (R2). An immediate consequence of (U2) is the following.

Lemma 3.1 *If an update operator \diamond satisfies (U2), and ψ is inconsistent, then $\psi \diamond \mu$ is inconsistent for any μ .*

The property above might appear undesirable: once an inconsistency is introduced in the knowledge base, there is no way to eliminate it. However, all we are saying is there is no way to eliminate it *by using update*. For example, revision does not have this behaviour; in fact, (R3) guarantees that the result of a revision is consistent provided that the new sentence introduced is itself consistent. This is another manifestation of the difference between update and revision. An inconsistent knowledge base is the result of an inadequate theory, and can be remedied with revision (or contraction) by adding new knowledge that supersedes the inconsistency (or removing contradictory knowledge using contraction). We can never repair an inconsistent theory using update, because update specifies a change in the world. If there is no set of worlds that fits our current description, we have no way of recording the change in the real world.

We drop rule (R6), and add instead three new postulates, (U6)~(U8). (U6) says that if updating a knowledge base with μ_1 guarantees μ_2 , and updating the same knowledge base with μ_2 guarantees μ_1 , then the two updates have the same effect. This is similar to condition (C7) in Gardenfors’s analysis of minimal changes of belief [5] and to conditional logic axiom CSO [17]. (U7) applies only to complete KB’s, in which there is no uncertainty over what are the possible worlds. If some possible world results from updating a complete KB with μ_1 and it also results from updating it with μ_2 , then this possible world must also result from updating the KB with $\mu_1 \vee \mu_2$. Finally, (U8) is what we called the “disjunction rule” in the Introduction. It guarantees that each possible world of the KB is given independent consideration.

The following lemma shows that we can obtain one direction of (R2) by using (U2) and (U8)

Lemma 3.2 If an update operator \diamond satisfies (U2) and (U8), then $\psi \wedge \mu$ implies $\psi \diamond \mu$.

However, as the example in the Introduction showed, update operators do not necessarily satisfy that $\psi \diamond \mu$ implies $\psi \wedge \mu$ when ψ is consistent with μ .

An interesting consequence of the postulates is *monotonicity*.

Lemma 3.3 If an update operator \diamond satisfies (U8), and ϕ implies ψ , then $\phi \diamond \mu$ implies $\psi \diamond \mu$.

Monotonicity has been deemed undesirable by the philosophers of theory revision. The reason is a result called “Gärdenfors’s impossibility theorem” [2, 6, 14], which shows that monotonicity is incompatible with postulates (R1)~(R4). More precisely, Theorem 7.10 of [6] implies that there is no non-trivial revision operator that satisfies monotonicity and (R1)–(R4). Since update operators do not satisfy (R2), this result does not apply to update.

Gärdenfors’s motivation in studying this problem is to use theory revision to define the conditional connective used in counterfactual reasoning. The idea is to use the *Ramsey Test*: interpret the conditional statement “given the state of the world described by ψ , if μ were true, then η would also be true” as $\psi \diamond \mu$ implies η . Intuitively, it would seem that this kind of statement is better modelled by using update instead of revision in the Ramsey Test. This intuition, together with the immunity of updates to Gärdenfors’s result, suggest further study of the connection between updates and conditional reasoning may be fruitful. Preliminary results are reported by Satoh *et al.* [18] and Grahne [8].

We can now formalize a notion of closeness between models that generalizes the particular measure used in the PMA. Instead of associating each KB with an ordering, let us consider a function that maps each interpretation I to a partial pre-order \leq_I . We say that this assignment is *faithful* if the following condition holds:

- For any $J \in \mathcal{I}$, if $I \neq J$ then $I <_I J$.

The following theorem shows that the postulates exactly capture all update operators defined by a partial pre-order. Unlike the case of revision, it turns out that the classes of operators defined by partial orders and partial pre-orders are the same.

Theorem 3.1 Let \diamond be an update operator. The following conditions are equivalent:

1. The update operator \diamond satisfies Conditions (U1)~(U8).
2. There exists a faithful assignment that maps each interpretation I to a partial pre-order \leq_I such

that

$$\text{Mod}(\psi \diamond \mu) = \bigcup_{I \in \text{Mod}(\psi)} \text{Min}(\text{Mod}(\mu), \leq_I).$$

3. There exists a persistent assignment that maps each interpretation I to a partial order \leq_I such that

$$\text{Mod}(\psi \diamond \mu) = \bigcup_{I \in \text{Mod}(\psi)} \text{Min}(\text{Mod}(\mu), \leq_I).$$

Proof Sketch. (1 \Rightarrow 2) We assign to each interpretation I a relation \leq_I defined as follows. For any interpretations J and J' , $J \leq_I J'$ if and only if either $J = I$ or $\text{Mod}(\text{form}(I) \diamond \text{form}(J, J')) = \{J\}$. We verify that Conditions (U1)~(U8) imply that this mapping is a faithful assignment such that

$$\text{Mod}(\psi \diamond \mu) = \bigcup_{I \in \text{Mod}(\psi)} \text{Min}(\text{Mod}(\mu), \leq_I).$$

(2 \Rightarrow 3) For a pre-order \leq_I , we define a relation \leq'_I as $J \leq'_I J'$ if and only if $J = J'$ or $J <_I J'$. It is easy to show that \leq'_I is a partial order and that $J <_I J'$ if and only if $J <'_I J'$. Hence, Statement 3 follows from Statement 2 by changing \leq_I to \leq'_I .

(3 \Rightarrow 1) Assume that there is a faithful assignment mapping each interpretation I to a partial order \leq_I . We define an update operator \diamond by

$$\text{Mod}(\psi \diamond \mu) = \bigcup_{I \in \text{Mod}(\psi)} \text{Min}(\text{Mod}(\mu), \leq_I).$$

We show that the update operator \diamond satisfies (U1)~(U8).

Comparing this result with Theorem 2.1, we see two differences between revision and update from a model-theoretic point of view. First, Theorem 3.1 refers to *partial* preorders while Theorem 2.1 uses *total* pre-orders. It turns out that a version of the revision postulates that accommodates partial preorders can be given, and we show this in [10]. It is also possible to design a class of update operators based on total pre-orders. If we replace (U6) and (U7) by postulate (U9) below, then we can prove the total pre-order analogue of Theorem 3.1. The proof is similar to that of Theorem 3.1, by defining, for any two interpretations J and J' , $J \leq_I J'$ if and only if either $J = I$ or $J \in \text{Mod}(\text{form}(I) \diamond \text{form}(J, J'))$.

(U9) If ψ is complete and $(\psi \diamond \mu) \wedge \phi$ is satisfiable then $\psi \diamond (\mu \wedge \phi)$ implies $(\psi \diamond \mu) \wedge \phi$.

It is worth pointing out that a total preorder associated with interpretation I is what Lewis [12] calls a *system of spheres centered at I*. Systems of spheres

play a central role in the semantics of Lewis's conditional logic; this brings up again the suggested connection between updates and conditional logic, which is explored further by Grahne [8].

The second and more important difference between revision and update is that, in the case of update, a different ordering is induced by each model of ψ , while for revision, only one ordering is induced by the whole of ψ . This "local" behaviour of update, contrasted with the "global" behaviour of revision, is essential to the difference between the two operators.

4 REASONING ABOUT ACTION

For the purposes of reasoning about action, the usual approach is to represent a particular action as a pair of a precondition and a postcondition. The precondition for the action encodes what the world must be like in order for the action to be executable. The postcondition describes the immediate consequences resulting from the action. Any update operator that satisfies our postulates can be used for reasoning about action by regarding postconditions for an action as new knowledge and by assuming that preconditions for the action are satisfied by the current KB. That is, the effect on KB ψ of performing action with precondition α and postcondition β will be ψ if ψ does not imply α , and $\psi \diamond \beta$ otherwise. Winslett [20] discusses how the frame, qualification and ramification problems are handled by this approach.²

Let us extend this idea by examining more closely what happens when ψ does not satisfy the precondition α . Presumably, the robot will return and report one of two outcomes: either α was true, and the action was carried out, or α failed and the action was not carried out. If we want a more elaborate model, we can also allow other outcomes, such as: α was true but the action could not be carried out for other reasons, or α could not be either verified or falsified. In each case, we can take advantage of the distinction between revision and update to incorporate into ψ all the information gained by the robot. For example, if the action was carried out, we can change the KB to $(\psi \diamond \alpha) \diamond \beta$. If the precondition was found false, we use $\psi \diamond \neg\alpha$. If the truth value of the precondition could not be determined, we use $\psi \bullet \alpha$ (contraction is discussed in the next section).

5 CONTRACTION AND ERASURE

Contraction is a change of belief or knowledge state induced by the loss of confidence in some sentence.

²Actually, Winslett uses for this purpose a variant of the PMA that orders interpretations in a way similar to the partial pre-order used in prioritized circumscription. Such variants are included in our class of update operators.

For example, if we believed that a paper was written by Turing, but new evidence has cast doubt on this belief, we contract the corresponding sentence from our knowledge base.

Alchourrón *et al.* [1] proposed rationality postulates for contraction. We denote by $\psi \bullet \mu$ a new knowledge base obtained from an old knowledge base ψ by contracting μ . The postulates for contraction, rephrased in our terms, are as follows.

- (C1) ψ implies $\psi \bullet \mu$.
- (C2) If ψ does not imply μ then $\psi \bullet \mu$ is equivalent to ψ .
- (C3) If μ is not a tautology then $\psi \bullet \mu$ is equivalent to ψ .
- (C3) If μ is not a tautology then $\psi \bullet \mu$ does not imply μ .
- (C4) If $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$ then $\psi_1 \bullet \mu_1 \equiv \psi_2 \bullet \mu_2$.
- (C5) $(\psi \bullet \mu) \wedge \mu$ implies ψ .

Alchourrón *et al.* [1] showed that contraction and revision are closely related: they proved that, given a revision operator \circ that satisfies (R1)~(R4), if we define a contraction operator \bullet by

$$\psi \bullet \mu \equiv \psi \vee (\psi \circ \neg\mu)$$

then the operator \bullet satisfies (C1)~(C5). Conversely, given a contraction operator \bullet that satisfies (C1)~(C4), if we define a revision operator \circ by

$$\psi \circ \mu \equiv (\psi \bullet \neg\mu) \wedge \mu$$

then the operator \circ satisfies (R1)~(R4).

We propose a new operator, *erasure*, which is to contraction as update is to revision. Erasing sentence μ from ψ means adding models to ψ ; for each model I , we add all those models closest to I in which μ is false. Intuitively, erasing μ means the world may have changed in such a way that μ is not true. In contrast, contracting μ means our description of the set of possible worlds must be adjusted to the possibility of μ being false.

The erasure operator \diamond for a given update operator \diamond is defined by

$$\psi \diamond \mu \equiv \psi \vee (\psi \diamond \neg\mu) \quad (U \rightarrow E).$$

This erasure operator satisfies the following postulates (E1)~(E5) and (E8) if the update operator satisfies (U1)~(U4) and (U8).

- (E1) ψ implies $\psi \diamond \mu$.
- (E2) If ψ implies $\neg\mu$ then $\psi \diamond \mu$ is equivalent to ψ .
- (E3) If ψ is satisfiable and μ is not a tautology then $\psi \diamond \mu$ does not imply μ .
- (E4) If $\psi_1 \equiv \psi_2$ and $\mu_1 \equiv \mu_2$ then $\psi_1 \diamond \mu_1 \equiv \psi_2 \diamond \mu_2$.

(E5) $(\psi \diamond \mu) \wedge \mu$ implies ψ .

(E8) $(\psi_1 \vee \psi_2) \diamond \mu$ is equivalent to $(\psi_1 \diamond \mu) \vee (\psi_2 \diamond \mu)$.

There are two differences between contraction and erasure in terms of postulates. One is that (E2) is weaker than (C2); since contraction of a sentence μ does not influence a KB ψ if ψ does not imply μ , but erasure of μ might modify ψ if ψ does not imply $\neg\mu$. The other one is that erasure needs the disjunctive rule (E8), but contraction does not.

Example 5.1 Consider Example 3.1 again. Recall we have a room with two objects in it, a book and a magazine, b means the book is on the floor, and m means the magazine is on the floor. The knowledge base ψ states that either the book is on the floor or the magazine is, but not both. Suppose that a contraction operator \bullet satisfies (C2). If we contract ψ by b then $\psi \bullet b$ is equivalent to ψ , since ψ does not imply b . This means that since the sentence that the book is on the floor is already questionable under ψ , contraction does not change ψ .

On the other hand, let an erasure operator \diamond be defined based on the PMA \diamond_{pma} . If we erase b from ψ then $\psi \diamond b$ is equivalent to $(b \wedge \neg m) \vee \neg b$. This can be interpreted as follows. ψ represents two possible worlds, M_1 and M_2 . In world M_1 , the book is on the floor but the magazine is not. Since b holds in M_1 , M_1 is altered to two worlds, M_1 itself and the world M_3 represented by $\neg b \wedge \neg m$, that is, neither the book nor the magazine is on the floor. In world M_2 , the magazine is on the floor but the book is not. Since b does not hold in M_2 , M_2 is retained as itself. Hence, $\psi \diamond b$ represents the three worlds, M_1 , M_2 and M_3 .

The intuitive difference between contraction and erasure can be explained in this example as follows. Contracting b means nothing has changed in the room, but if the KB believes that the book is on the floor, make sure this belief is retracted. Since the KB has no such belief, the contraction has no effect. Erasing b means the state of the room has changed in such a way that, if the book was on the floor before, it has now been moved in an unpredictable way. This affects only those possible worlds in which the book was on the floor. The result is that we can no longer deduce anything about the location of the magazine from the fact that the book is not on the floor.

There is another operation which appears perhaps more natural than erasure. Suppose the state of the room has changed in such a way that the location of the book is now unpredictable, and we want to reflect this change in the knowledge base. We formalize this operation, called *symmetric erasure*, after the Theorem below.

The following theorem gives a correspondence between update and erasure similar to the correspondence be-

tween revision and contraction.

Theorem 5.1

1. If an update operator \diamond satisfies (U1)~(U4) and (U8), then the erasure operator \diamond defined by $(U \rightarrow E)$ satisfies (E1)~(E5) and (E8).

2. If an erasure operator \diamond satisfies (E1)~(E4) and (E8), then the update operator \diamond defined by

$$\psi \diamond \mu \equiv (\psi \diamond \neg \mu) \wedge \mu \quad (E \rightarrow U)$$

satisfies (U1)~(U4) and (U8).

3. Suppose that an update operator \diamond satisfies (U1)~(U4) and (U8). Then, we can define an erasure operator by $(U \rightarrow E)$. The update operator obtained from the erasure operator by $(E \rightarrow U)$ is equal to the original update operator \diamond .

4. Suppose that an erasure operator \diamond satisfies (E1)~(E5) and (E8). Then, we can define an update operator by $(E \rightarrow U)$. The erasure operator obtained from the update operator by $(U \rightarrow E)$ is equal to the original erasure operator \diamond .

Winslett [21] discusses an operator called *Forget*, which she compares with contraction. It turns out that *Forget*, given an update operator \diamond , is equivalent to

$$(\psi \diamond \mu) \vee (\psi \diamond \neg \mu).$$

We call this operator *symmetric erasure* because μ and its negation play the same role in its definition. The main difference between erasure and symmetric erasure is that erasure does not affect the possible worlds in which $\neg \mu$ holds, but symmetric erasure does. Going back to Example 5.1, the symmetric erasure of b from ψ reflects the fact that someone has picked up the book and unpredictably decided to place it on the floor or on the table. The result of this symmetric erasure is the knowledge base with no information, since there is nothing we can say about either the book or the magazine after this change.

We can show similar postulates for symmetric erasure to those for erasure, and prove a similar theorem to Theorem 5.1. A natural definition of *symmetric contraction* follows from the above discussions, and similar results can be shown for it. Gärdenfors [6] defines an operator similar to symmetric contraction, which he calls *complete contraction*, and proposes to use it to model “even if” conditionals.

6 CONCLUSION

The distinction between update and revision is an important one, and it has been overlooked in the literature since it was pointed out by Keller and Winslett [12]. We have formalized this distinction and

given a model-theoretic characterization of updates in terms of orderings among interpretations. We have defined and characterized *erasure*, which is to update as contraction is to revision.

Many problems remain to explore. The connection between updates and conditional logic is one being pursued by several researchers [18, 8]. Another is computational tractability of updates and erasures. A third is the combined use of different theory change operators—revision, contraction, update, erasure—in specific applications, as suggested in Section 4.

Acknowledgements

We thank Ken Satoh for suggesting rule (U6). The second author thanks the Natural Sciences and Engineering Research Council of Canada and the Institute for Robotics and Intelligent Systems for their support.

References

- [1] Carlos E. Alchourrón, Peter Gärdenfors and David Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.
- [2] Horacio L. Arló Costa. Conditionals and monotonic belief revisions: the success postulate. In *VI Simposio Latinoamericano de Lógica Matemática*, August 1989.
- [3] Mukesh Dalal. Investigations into a theory of knowledge base revision: Preliminary Report. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 475–479, 1988.
- [4] Peter Gärdenfors. Conditionals and changes of belief. *Acta Philosophica Fennica*, Vol. XXX(1978), 381–404.
- [5] Peter Gärdenfors. An Epistemic Approach to Conditionals. *American Philosophical Quarterly* Vol. 18, Number 3, July 1981, 203–211.
- [6] Peter Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Bradford Books, MIT Press, Cambridge, MA, 1988.
- [7] M.L. Ginsberg and D.E. Smith. Reasoning about action I: a possible worlds approach. In *Readings in Nonmonotonic Reasoning*, M.L. Ginsberg, ed., Morgan Kaufmann, Los Altos, 1987.
- [8] Gösta Grahne. Updates and counterfactuals. In Allen, J.A., Fikes, R., and Sandewell, E. (eds.) *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*. San Mateo, California: Morgan Kaufmann. To appear in 1991.
- [9] Hirofumi Katsuno and Alberto O. Mendelzon. A unified view of propositional knowledge base updates. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1413–1419, 1989.
- [10] Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. Technical Report KRR-TR-90-3, Department of Computer Science, University of Toronto, March 1990. Revised version to appear in *Artificial Intelligence*.
- [11] Arthur M. Keller and Marianne Winslett Wilkins. On the use of an extended relational model to handle changing incomplete information. *IEEE Trans. on Software Engineering*, SE-11:7, pages 620–633, July 1985.
- [12] D. K. Lewis. *Counterfactuals*. Blackwell, Oxford, 1973.
- [13] David Makinson. How to give it up: A survey of some formal aspects of the logic of theory change. *Synthèse*, 62:347–363, 1985.
- [14] David Makinson. The Gärdenfors impossibility theorem in non-monotonic contexts. Unpublished. To appear in *Studia Logica*.
- [15] Bernhard Nebel. A knowledge level analysis of belief revision. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 301–311, 1989.
- [16] Donald Nute. Conditional logic. In Dov M. Gabbay and Franz Guenther, editors, *Handbook of Philosophical Logic*, D. Reidel, Dordrecht, 1984.
- [17] Anand S. Rao and Norman Y. Foo. Minimal change and maximal coherence: A basis for belief revision and reasoning about action. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 966–971, 1989.
- [18] Ken Satoh, Hirofumi Katsuno and Alberto O. Mendelzon. A unified view of consequence relations, belief revision, and conditional logic. Unpublished manuscript.
- [19] Marianne Winslett. Reasoning about action using a possible models approach. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 89–93, 1988.
- [20] Marianne Winslett. Theory revision semantics for use in reasoning about action. Unpublished manuscript, June 1989.
- [21] Marianne Winslett. *Updating Logical Databases*. Cambridge University Press, in press.

Modal nonmonotonic logics: ranges, characterization, computation

Wiktor Marek

Department of Computer Science
University of Kentucky
Lexington, KY 40506-0027

Grigori F. Shvarts

Program Systems Institute
USSR Academy of Sciences
152140 Pereslavl-Zalesky, USSR

Mirosław Truszczyński

Department of Computer Science
University of Kentucky
Lexington, KY 40506-0027

Abstract

In the paper, we investigate the way in which nonmonotonic modal logics depend on their underlying monotonic modal logics. Most notably, we study when different monotonic modal logics define the same nonmonotonic system. In particular, we show that for an important class of the so called *stratified* theories all nonmonotonic logics considered in the paper, with the exception of S5, coincide. It turns out that in some cases, nonstandard (that is, non-normal) logics have interesting nonmonotonic counterparts. Two such systems are investigated in the paper in detail. For the case of finite theories, all nonmonotonic logics considered are shown to be decidable and an appropriate algorithm is presented.

1 INTRODUCTION

Many types of commonsense reasonings can be faithfully represented within modal nonmonotonic logic. These reasonings include: default logic of Reiter [Rei80], nonmonotonic logic of belief of Moore [Moo85], truth maintenance of Doyle [Doy79], and some important aspects of logic programming [GL88]. Therefore, in this paper we will study three fundamental issues of modal nonmonotonic logics:

- (a) dependence of such logics on the underlying monotonic logic,
- (b) characterization of expansions,
- (c) computation of expansions.

Most known modal nonmonotonic logics belong to the family proposed by McDermott [McD82]. In this paper we focus on these logics only. The reader is referred to [Kam90], [Tru90], [Sho87] and [LS90] for the discussion of other modal nonmonotonic systems.

There is a rich variety of different monotonic modal logics [Che80], [HC84]. An important question is: how

much of this variety carries over to the nonmonotonic case. The most important and somewhat unexpected result of our paper is that the structure of the family of nonmonotonic modal logics is much simpler. Speaking more formally, it is often the case that different monotonic modal logics collapse to the same nonmonotonic system.

The results of this paper focus on modal logics whose nonmonotonic counterparts are applicable in knowledge representation. We identified three such logics: K45, N and W5. It turns out that for each of these logics there is a whole family of monotonic modal logics that generate the same system in the nonmonotonic case. For nonmonotonic logics K45, N and W5, as well as many others, we characterize expansions and provide algorithms for computing them.

Throughout the paper, by Cn we denote the operator of propositional consequence, and by $Cn_{\mathcal{S}}$, where \mathcal{S} stands for a modal logic, the operator of provability in logic \mathcal{S} . Our further discussion assumes familiarity with basic notions of modal logics as given in [HC84] and [Che80]. The mechanism of nonmonotonic modal consequence operation based on the (monotonic) modal logic \mathcal{S} has been introduced in [MD80] and [McD82]. This method can be shortly presented as follows: Given a modal logic \mathcal{S} and a theory I in the language with one modal operator L , denoted \mathcal{L}_L , a theory $T \subseteq \mathcal{L}_L$ is called an \mathcal{S} -expansion of I if it satisfies the equation:

$$T = Cn_{\mathcal{S}}(I \cup \{\neg L\varphi : \varphi \notin T\}), \quad (1)$$

With the interpretation of modal operator L as “is known” or “is believed”, T is an expansion of I if T is precisely the collection of these formulas which can be derived in \mathcal{S} from I and statements about “ignorance” or “negative introspection” with respect to T . This circular aspect and self-reference of the concept of expansion results in the fact that the equation (1) may have single or multiple solutions, and sometimes even no solution. Once all such “points of view” T are identified, we compute the nonmonotonic consequence of I in \mathcal{S} as the intersection of all \mathcal{S} -expansions of I .

Once the nonmonotonic counterparts for modal logics are defined, the following question becomes of fundamental importance: **Which modal nonmonotonic logics are applicable in knowledge representation?** Early attempts to identify such logics have been unsuccessful. The case of \mathcal{S} equal to propositional calculus (in the language \mathcal{L}_L) leads to counterintuitive expansions, for example, containing both p and $\neg Lp$. The other extreme, $\mathcal{S} = \mathbf{S5}$, collapses to monotonic **S5** that is, the resulting nonmonotonic consequence operator coincides with the monotonic **S5**, [McD82], [MT90b].

In a reaction to the above mentioned failures [Moo85] introduced a seemingly different scheme:

$$T = Cn(I \cup \{\varphi \Rightarrow L\varphi : \varphi \in T\} \cup \{\neg L\varphi : \varphi \notin T\}) \quad (2)$$

and argued that the solutions to the equation (2) better capture the intuitions associated with the states of belief of a fully introspective agent than the general scheme (1). This scheme is very specialized — notice the absence of modal parameter \mathcal{S} in the equation (2). Yet, it turns out that for consistent theories T , Moore's expansions, called *autoepistemic* or *stable* expansions, coincide with **K45**-expansions [Shv90]. In other words, the logic of Moore belongs to the family of nonmonotonic formalisms definable by the scheme (1).

One of the most widely studied and used nonmonotonic formalisms is the default logic of Reiter [Rei80]. Intensive studies were undertaken to find a modal counterpart of the default logic [Kon88], [MT89a], [MT90b], [LS90]. It turned out that under the translation that assigns to a default rule $\frac{\alpha: \beta_1 \dots \beta_n}{\gamma}$ a modal formula: $L\alpha \wedge \neg LL\neg\beta_1 \wedge \dots \wedge \neg LL\neg\beta_n \Rightarrow \gamma$, extensions of default theories can be faithfully described as **N**-expansions, where **N** is the modal logic of necessitation, that is the modal logic without any scheme for handling modalities [MT90b]. In addition, the same modal logic can be used to represent stable semantics for logic programs, as well as ordinary truth maintenance systems ([MT89b]).

Third nonmonotonic modal logic with natural applications in knowledge representation is the modal nonmonotonic logic associated with the modal logic **W5**, which contains one axiom schema:

$$\text{W5: } \neg L\neg L\varphi \Rightarrow (\varphi \Rightarrow L\varphi),$$

a weaker variant of 5. It is shown in [MT90b] that the nonmonotonic logic **W5** allows to provide a natural semantics for two important modes of nonmonotonic reasoning: logic programming with classical negation (cf. [GL90]), and truth maintenance systems in which we admit rules containing literals (not only atoms).

The nonmonotonic logic **W5** possesses a semantic characterization similar to Moore's characterization of

stable (that is, **K45**-) expansions. The following fixpoint equation characterizes **W5**-expansions [MT90b]:

$$T = Cn(I \cup \{\varphi \Rightarrow L\varphi : \varphi \in T\} \cup \{\neg L\varphi : \varphi \notin T\}). \quad (3)$$

This fixpoint equation relaxes Moore's definition of *autoepistemic valuation with index T*. Recall that V is an *autoepistemic valuation with index T* if $V(L\varphi) = 1$ precisely when $\varphi \in T$. Here, the class of valuations V contains, as before, valuations that satisfy condition $V(L\varphi) = 0$ if $\varphi \notin T$. But, on positive side, for $\varphi \in T$, we relax the condition as follows: we require that $V(\varphi \Rightarrow L\varphi) = 1$. All autoepistemic valuations satisfy this condition, but there are other valuations which need to be considered as well. The intuition here is that we want V to evaluate $L\varphi$ as 1 *providing* that V evaluates φ as 1. Since we extended the class of valuations under consideration, a stronger condition is imposed on a fixpoint. Any solution of equation (3) is called a *strict expansion* of I . The important point here is that such strict expansions are definable semantically, and in a natural fashion.

Summarizing our discussion, three logics **K45**, **N** and **W5** seem to be of particular interest in knowledge representation. These three logics have several puzzling things in common. First of all, they are clearly off the main research track of classical modal logic. To our knowledge, even **K45** was very little studied, even though it has an elegant and well understood Kripke semantics. Secondly, two of the logics — **N** and **W5** are subnormal that is, do not satisfy axiom schema K. The third one, **K45** is normal but, as proved in [MT90b], it is equivalent to the subnormal modal logic 5 satisfying only axiom schema 5, in the sense that both logics have the same “nonmonotonic variant”.

The meaning of this last result (collapse of nonmonotonic modal logics 5 and **K45**) was deeply intriguing; it indicated that *different* monotonic logics may generate the same notion of (consistent) expansion, and consequently the same nonmonotonic consequence operation! In other words, the realm of nonmonotonic modal logics is much less diversified than that of monotonic modal logics. This, in turn, means that in the nonmonotonic case the axioms to manipulate modality play a different role than in the monotonic case.

On a closer inspection these observations seem to be less puzzling. After all, nonmonotonic \mathcal{S} -consequences of a theory I are often strictly larger than monotonic \mathcal{S} -consequences of I due to the powerful principle of “negation as failure to prove” which allows us to use in the reasonings formulas expressing negative introspection ($\{\neg L\varphi : \varphi \notin T\}$). Precisely speaking, let $\mathcal{S}_1 \subseteq \mathcal{S}_2 \subseteq \mathbf{S5}$ be two modal logics. If for every stable theory $T \subseteq \mathcal{L}_L$, and for every instance ψ of any axiom schema of \mathcal{S}_2 , $\{\neg L\varphi : \varphi \notin T\} \vdash_{\mathcal{S}_1} \psi$, then nonmonotonic logics \mathcal{S}_1 and \mathcal{S}_2 are identical.

A number of basic questions arise on the dependence

of the nonmonotonic \mathcal{S} -consequences on the underlying logic \mathcal{S} . The first question is this: Are there nonmonotonic logics different from the autoepistemic logic of Moore that can be equivalently defined through McDermott's scheme (1) for different monotonic logics \mathcal{S} ? If so, what are the properties of these logics and their mutual relationships. Although we do not have complete answers to these questions, it turns out that a number of nontrivial facts can be proved about "ranges" of modal logics collapsing to the same nonmonotonic logic. We shall give a number of such results in this paper.

Formally, a *range* is a collection of monotonic modal logics generating the same concept of a consistent expansion. A number of ranges will be exhibited below namely, for the logics **N**, **W5** and **K45**. Important open questions are: Are there any trivial (one-element) ranges? Are there any other nontrivial ranges than those exhibited in this paper? One needs to point that the fact that nontrivial ranges exist is, potentially, quite beneficial. For instance a range may contain very different logics. Some may have a nice automated theorem proving mechanism, whereas others may have an elegant semantics. In such case, for the *nonmonotonic* logic associated with the range, we might have completely different mechanisms for semantical and for syntactical manipulations. In fact we may have completely different semantics, all "nonmonotonically" equivalent. This is the case for the range associated with Moore's logic. The freedom of selecting different theorem provers may turn out useful in practical implementations.

In addition, it turns out that syntactic restrictions on I make some ranges coagulate. Two important types of syntactic restrictions we study are stratification and restriction to formulas with negative introspection only. In each case, the ranges of equivalent (nonmonotonically) modal logics are exhibited.

Our results on the ranges can be summarized as follows:

- General theories: Logics **N**, **W5**, **K45** possess nontrivial, ranges.
- Theories consisting of formulas with negative introspection only: A wide range from **N** to **KD45**.
- Stratified theories: All known ranges collapse to one.

Next, we consider the problem of characterization of \mathcal{S} -expansions. Several general results were obtained in [Shv90]. In particular, they imply characterizations of \mathcal{S} -expansions for several normal modal logics including **K**, **T**, **S4**, **K45**. We strengthened the results of [Shv90] and derived characterizations of **N**-expansions and **W5**-expansions. All characterizations involve modal atoms appearing as subformulas in formulas of a theory whose expansions we study.

Most importantly, the characterizations we obtained form the basis for the third component of our investigation, namely computation of \mathcal{S} -expansions. We present algorithms for computing \mathcal{S} -expansions for a large class of logics (including **N**, **W5** and **K45**). We illustrate the algorithms with an example.

The paper is organized as follows. The next section contains our results on the ranges both for the case of general theories, as well as those subject to some syntactic restrictions. The results are gathered at the end of the section in appropriate diagrams. Section 3 discusses characterizations of expansions and algorithms for their computation. Section 4 contains conclusions.

2 \mathcal{S} -EXPANSIONS — RANGES

Because of space limitations, we will assume some familiarity with the results of [Moo85], [Kon89] and [MT89a]. In addition to standard modal logics such as **K**, **T**, **S4**, **S5**, **K45** etc., we shall consider logic **N** of necessitation (with no axiom scheme at all) and the following two schemes: **W5** (introduced above) and:

WK: $L(\varphi \Rightarrow \psi) \wedge L\varphi \Rightarrow \neg L\neg L\psi$ (a variant of **K** that, given $L\varphi$ and $L(\varphi \Rightarrow \psi)$, instead asserting $L\psi$ as **K** does, asserts $\neg L\neg L\psi$).

We start with investigating the general properties of \mathcal{S} -expansions. First, let us mention several facts concerning *stable* sets [Sta80, McD82, Moo85, Mar89]. For each theory $S \subseteq \mathcal{L}$ there is a unique stable set T such that $T \cap \mathcal{L}$ is exactly the set of logical consequences of S . This unique stable set will be denoted by $E(S)$. A constructive definition of $E(S)$, for a finite S , is given in [Mar89]. In fact, it can be shown, that

$$E(S) = Cn_{\mathbf{S5}}(S \cup \{\neg L\varphi : \varphi \in \mathcal{L} \setminus Cn(S)\}).$$

Clearly, if a modal logic \mathcal{S} contains the necessitation rule then each \mathcal{S} -expansion is stable.

Let us now pass on to investigations of mutual relationships between classes of \mathcal{S} -expansions for various logics \mathcal{S} . We have the following simple result.

Proposition 2.1 ([McD82]) *Let \mathcal{S} and \mathcal{T} be two modal logics, $S \subseteq T \subseteq S5$. Then each \mathcal{S} -expansion of I is a \mathcal{T} -expansion of I .*

The following three theorems show that the notion of range is nontrivial. First result, obtained in [MT90b] shows that the autoepistemic logic of Moore can be defined not only by means of the logic **K45** or **KD45**, which was known already to Konolige [Kon88], but by any logic in a much wider class. In fact, our result shows that the crucial role in the autoepistemic logic is played by the axiom schema 5, which allows to derive positive introspection from the negative introspection.

Theorem 2.2 Let \mathcal{S} be a modal logic, $5 \subseteq \mathcal{S} \subseteq \text{KD45}$. Let $I, T \subseteq \mathcal{L}_L$. The theory T is a stable expansion of I if and only if T is an \mathcal{S} -expansion of I .

Thus, for the whole range of logics between 5 and KD45 the same notion of expansion is obtained. A similar result holds for N-expansions.

Theorem 2.3 Let \mathcal{S} be a modal logic, $N \subseteq \mathcal{S} \subseteq \text{WK}$. Let $I, T \subseteq \mathcal{L}_L$. The theory T is an N-expansion of I if and only if T is an \mathcal{S} -expansion of I .

Our final result of that type concerns the case of strict expansions. Firstly, it shows that strict expansions can also be characterized by means of scheme (1). Secondly, it exhibits the whole range of logics that can be used for that purpose.

Theorem 2.4 Let \mathcal{S} be a modal logic, $\text{W5} \subseteq \mathcal{S} \subseteq \text{D4W5}$. Let $I, T \subseteq \mathcal{L}_L$. Then, the theory T is a strict expansion of I if and only if T is an \mathcal{S} -expansion of I .

Theorems 2.2, 2.3 and 2.4 give rise to an interesting theoretical problem. Let \mathcal{S} be a monotonic modal logic. By $R(\mathcal{S})$ we denote the range of all monotonic modal logics $T \subseteq \text{S5}$ “nonmonotonically equivalent” to \mathcal{S} . A general question that arises is: What is the structure of the set $R(\mathcal{S})$? It is easy to see that if $T_1, T_2 \in R(\mathcal{S})$, then for every modal logic T such that $T_1 \subseteq T \subseteq T_2$ we have $T \in R(\mathcal{S})$. Thus, each $R(\mathcal{S})$ is an interval in the partial ordering of modal logics by inclusion relation. The following question is still open: Does each such interval have the least element, the greatest element? Another general question is: Does each interval $R(\mathcal{S})$ always contain logics other than \mathcal{S} ?

We have already seen that for the whole ranges of logics \mathcal{S} , the notions of expansions defined by these logics are equivalent (Theorems 2.2, 2.3 and 2.4). Below we show that if we restrict the class of theories I , even stronger results hold. Namely, we have the following two theorems. First of them was proved in [Shv90], the second one in [MT90b].

Theorem 2.5 Let $I \subseteq \mathcal{L}$ be consistent. For each logic \mathcal{S} such that

- (a) $N \subseteq \mathcal{S}$, and
 - (b) $\mathcal{S} \subseteq \text{KD45}$ or $\mathcal{S} \subseteq \text{S4}$,
- the theory $E(I)$ is the only \mathcal{S} -expansion of I .

Theorem 2.5 says that there is a big range of modal logics in the case of objective theories.

Theorem 2.6 Let \mathcal{S} be a modal logic contained in KD45. If I consists only of modal clauses with negative introspection, then T is an \mathcal{S} -expansion of I if and only if T is an N-expansion of I .

Theorem 2.5 can be generalized to a wider class of theories. A theory I is *strongly stratified* if

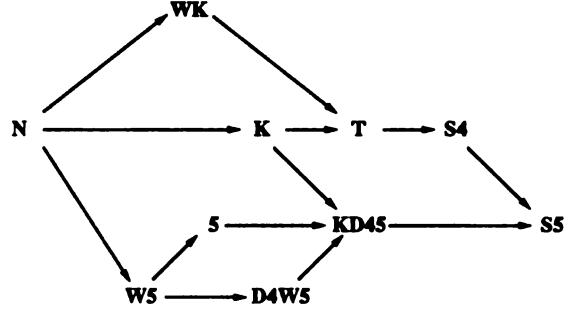


Figure 1: The hierarchy of modal logics

STRAT1 Each formula in I is of the form $a(\varphi) \Rightarrow c(\varphi)$, $c(\varphi) \in \mathcal{L}$, for $\varphi \in I$, and $\{c(\varphi) : \varphi \in I\}$ is consistent.

STRAT2 I has a partition (called *stratification*) $I = I_1 \cup \dots \cup I_n$ into disjoint and nonempty sets such that for each $\varphi \in I_j$, $1 \leq j \leq n$, propositional letters occurring in $c(\varphi)$ do not occur in formulas of $I_1 \cup \dots \cup I_{j-1}$ and do not occur in formulas of I_j under the scope of the operator L .

This notion of stratification is closely related to stratification of logic programs, as introduced in [ABW87] and extends the concept of stratification as introduced in [Gel87]. The difference is that where Gelfond requires that the consequents $c(\varphi)$ of formulas in I are disjunctions of atoms, we impose a weaker condition, namely that these consequents form a consistent theory. We will see that as long as we restrict ourselves to strongly stratified theories all ranges collapse into one large range. This is the subject of the next theorem.

Theorem 2.7 Let $I \subseteq \mathcal{L}$ be strongly stratified. For each logic \mathcal{S} such that

- (a) $N \subseteq \mathcal{S}$, and
- (b) $\mathcal{S} \subseteq \text{KD45}$ or $\mathcal{S} \subseteq \text{S4}$,

the theory I has exactly one \mathcal{S} -expansion and for all these logics \mathcal{S} , \mathcal{S} -expansions of I coincide.

We collect now the results on the relationships among the classes of expansions discussed in the paper. We considered the following logics: N , WK , K , T , S4 , S5 , W5 , D4W5 , 5 , KD45 . Inclusion relation diagram for these logics is shown in Figure 1. An arrow in the diagram directed from a logic \mathcal{S}_1 to a logic \mathcal{S}_2 indicates that $\mathcal{S}_1 \subseteq \mathcal{S}_2$.

All of these inclusions are straightforward. To see that $\text{WK} \subseteq T$ simply check that the axiom schema WK holds in every T-Kripke model (that is, in every Kripke model with a reflexive admissibility relation).

According to Proposition 2.1 (b), the same inclusion relations hold for classes of \mathcal{S} -expansions. In addition, for each theory I , a WK -expansion of I is also

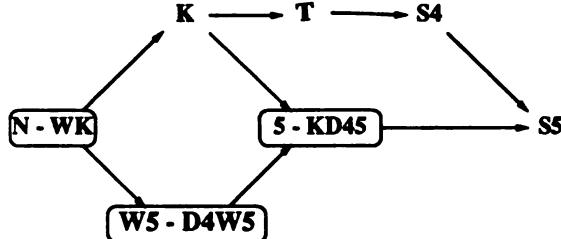


Figure 2: The hierarchy of modal nonmonotonic logics

an \mathbf{N} -expansion of I (Theorem 2.3) and, therefore, a \mathbf{K} -expansion of I . Theorems 2.2, 2.3 and 2.4 state that logics \mathbf{N} and \mathbf{WK} , $\mathbf{5}$ and $\mathbf{KD45}$, and $\mathbf{W5}$ and $\mathbf{D4W5}$, respectively, define the same notion of expansion. All these properties are summarized in the diagram in Figure 2. The arrow between the symbols for two logics S and T indicates that for each theory I , every S -expansion of I is a T -expansion of I . The ranges of logics that define the same notion of expansion are indicated by ovals. In all other cases, there are theories, for which different logics define different classes of expansions. This is illustrated by examples given in Section 3.

Thus, the only generally true inclusion relationships between the classes of expansions shown in Figure 2 are those indicated by arrows.

3 CHARACTERIZATIONS OF S -EXPANSIONS AND ALGORITHMS

Continuing our program, we move to characterize S -expansions. In the case of many modal logics including logics \mathbf{K} , \mathbf{T} , $\mathbf{S4}$, $\mathbf{K45}$, the problem was solved by Shvarts [Shv90]. Below we strengthen his results and extend them to logics \mathbf{N} and $\mathbf{W5}$.

By I^L we denote the set of all subformulas of the formulas from I of the form $L\psi$. Let T be a consistent stable theory containing I . Let $\Psi = I^L \cap T$, and $\Phi = I^L \setminus \Psi$. We denote by $\neg\Phi$ the set $\{\neg\varphi : \varphi \in \Phi\}$. Obviously, if $L\psi \in T$ then $\psi \in T$, and if $L\varphi \in \Phi$ then $\varphi \notin T$ and $\neg L\varphi \in T$. Hence $I \cup \neg\Phi \cup \Psi \cup \{\psi : L\psi \in \Psi\}$ is consistent and is contained in T . Furthermore, for every $L\varphi \in \Phi$, $\varphi \notin Cn(I \cup \neg\Phi \cup \Psi \cup \{\psi : L\psi \in \Psi\})$. These observations motivate the following definition. Let $\Phi \subseteq I^L$, $\Psi = I^L \setminus \Phi$. Φ is said to be *admissible* for I iff $I \cup \neg\Phi \cup \Psi \cup \{\psi : L\psi \in \Psi\}$ is propositionally consistent and for each $L\varphi \in \Phi$, $\varphi \notin Cn(I \cup \neg\Phi \cup \Psi \cup \{\psi : L\psi \in \Psi\})$. Given a modal logic S , we call a set $\Phi \subseteq I^L$ *S -admissible* if Φ is admissible for I and for each $L\psi \in \Psi$, $I \cup \neg\Phi \vdash_S \psi$.

In what follows, the stable set generated by $Cn(I \cup \neg\Phi \cup \Psi \cup \{\psi : L\psi \in \Psi\}) \cap \mathcal{L}$ plays a special role. We will denote it by

$$T_{I,\Phi} = E(Cn(I \cup \neg\Phi \cup \Psi \cup \{\psi : L\psi \in \Psi\}) \cap \mathcal{L}).$$

The role of the stable set $T_{I,\Phi}$ is explained by the following result of Shvarts [Shv90].

Theorem 3.1 *Let S be any modal logic contained in $S5$, and let Φ be S -admissible for I . Then $T_{I,\Phi}$ is an S -expansion of I .*

We will first extend general results of [Shv90] to a wider class of logics. A class of frames C is *closed* if for all (M_1, R_1) , $(M_2, R_2) \in C$ such that $M_1 \cap M_2 = \emptyset$, the frame $(M_1 \cup M_2, R_1 \cup (M_1 \times M_2) \cup R_2)$ belongs to C . In the next theorem, we need yet another concept. If F is a complete frame, that is $F = (M, M \times M)$, then by *one-element extension* (1-extension) of F we mean a frame of the form $(M \cup \{a\}, (\{a\} \times M) \cup M \times M)$ or $(M \cup \{a\}, (\{a\} \times M) \cup \{a\} \times M) \cup M \times M$, where $a \notin M$. Thus, 1-extension of the complete frame F adds to F one new world a from which all the worlds of F are accessible. In addition a may be accessible from itself or not. By the (1-ext) property of a logic S we mean the following property:

(1-ext) For any complete frame F , at least one 1-extension G of F is a frame for S .

Theorem 3.2 *Let $S \subseteq S5$ be a modal logic.*

(a) *If S has the property (1-ext) then for each set Φ that is S -admissible for a theory I , $T_{I,\Phi}$ is the only consistent S -expansion of I containing $I \cup \neg\Phi$.*

(b) *If S has the property (1-ext) and, in addition, S is characterized by a closed class of frames C , then T is an S -expansion of I if and only if $T = T_{I,\Phi}$, for some set Φ which is S -admissible for I .*

This theorem is a slight extension of Theorems 3.2 and 3.3 from [Shv90]. A minor modification of the original arguments from [Shv90] can be used to prove it. We omit the details here.

Theorem 3.2 (a) applies to the logics \mathbf{N} , $\mathbf{W5}$, \mathbf{K} , \mathbf{T} , $\mathbf{KD45}$ and $\mathbf{S4}$, and Theorem 3.2 (b) applies to logics \mathbf{K} , \mathbf{T} and $\mathbf{S4}$. In this paper, using different methods, we show that its statement holds also for the logic \mathbf{N} . All these results are gathered in the following proposition.

Theorem 3.3 (a) *Let S be any of \mathbf{N} , \mathbf{K} , \mathbf{T} , $\mathbf{S4}$, $\mathbf{K45}$, and let Φ be S -admissible for I . Then $T_{I,\Phi}$ is the unique S -expansion of I containing $I \cup \neg\Phi$.*

(b) *Let S be any of \mathbf{N} , \mathbf{K} , \mathbf{T} , $\mathbf{S4}$. Then, T is an S -expansion of I if and only if $T = T_{I,\Phi}$, for some S -admissible Φ .*

Theorem 3.3(b) does not hold for logics $\mathbf{K45}$ and $\mathbf{W5}$. For example, theory $I = \{Lp \Rightarrow p\}$ has a stable expansion $T_{I,\emptyset}$, but \emptyset is not $\mathbf{K45}$ -admissible for I . To

deal with the case of logic **K45** we need the following notion. A set Φ , admissible for I , is said to be *propositionally admissible for I* if for each $L\psi \in \Psi$, $I \cup \neg\Phi \cup \Psi \vdash \psi$. Shvarts [Shv88, Shv90] proved the following characterization of **K45**-expansions (stable expansions).

Theorem 3.4 *T is a stable expansion (that is, a **K45**-expansion) of I if and only if T is $T_{I,\Phi}$ for some Φ that is propositionally admissible for I .*

We apply here a similar technique to deal with the case of strict expansions that is, **W5**-expansions. Let $\Phi \subseteq I^L$, $\Psi = I^L \setminus \Phi$. The set Φ is said to be *strictly admissible for I* , if Φ is admissible for I and for each $L\psi \in \Psi$, $I \cup \neg\Phi \cup \{\varphi \Rightarrow L\varphi : L\varphi \in \Psi\} \vdash \psi$.

Theorem 3.5 *If T is consistent, then T is a strict expansion (that is, a **W5**-expansion) of I if and only if $T = T_{I,\Phi}$ for some Φ being strictly admissible for I .*

Now we are in a position to show that the inclusions between classes of \mathcal{S} -expansions shown in Figure 2 are the only ones that hold in general.

Example:

1. Let $I_1 = \emptyset$. Each stable theory is an **S5**-expansion of I_1 . Not every stable theory is an **S4**-expansion of I_1 , or a stable expansion (**KD45**-expansion) of I_1 . Thus, in general, the notions of **S5**-expansion and **S4**-expansion, and of **S5**-expansion and **KD45**-expansion are different.
2. Let $I_2 = \{L(Lp \Rightarrow LLp) \Rightarrow p\}$. It is easy to see that $E(p)$ is an **S4**-expansion of I_2 (in fact, the only **S4**-expansion of I_2). By Theorem 3.3 (b), $E(p) = T_{I_2,\Phi}$, where $\Phi = I_2^L \setminus (I_2^L \cap E(p))$. Clearly, in our case, $\Phi = \emptyset$. Consider the **T**-model $\mathcal{M} = (M, R, W)$, where $M = \{a, b, c\}$, $R = \{(a, a), (b, b), (c, c), (a, b), (b, c), (c, a)\}$, and $W(a) = W(b) = \{p\}$ and $W(c) = \emptyset$. It is easy to see that $\mathcal{M} \models I_2$ but $\mathcal{M} \not\models p$. Thus, $I_2 \not\models_T p$. Consequently, $\Phi = \emptyset$ is not **T**-admissible for I_2 and $E(p)$ is not a **T**-expansion of I_2 .
3. Let $I_3 = \{L(LLp \wedge L(Lp \Rightarrow p) \Rightarrow Lp) \Rightarrow p\}$. Using similar methods as before, we show that **K**-expansions are not, in general, **W5**-expansions.
4. Theory $I_4 = \{L(Lp \Rightarrow p) \Rightarrow p\}$ shows that **T**-expansions are not **K**-expansions.
5. Theory $I_5 = \{L(\neg L \neg Lp \Rightarrow (p \Rightarrow Lp)) \Rightarrow p\}$ shows that **W5**-expansions are not, in general, **S4**-expansions, and that **KD45**-expansions are not, in general, **S4**-expansions.
6. Let $I_6 = \{Lp \Rightarrow p\}$. Put $T = E(p)$. It is easy to see that $\Phi = \{Lp\}$ is propositionally admissible and that $T_{I_6,\Phi} = E(p)$. Thus, T is a **K45**-expansion of I_6 . Consider a valuation v of \mathcal{L}_L such that $v(p) = 0$, and $v(L\varphi) = 1$ if and only if $\varphi \neq p$ and $\varphi \in T$. Then, $v(I_6 \cup \{\varphi \Rightarrow L\varphi : \varphi \in T\}) = 1$ and $v(p) = 0$. But then, T is not a **W5**-expansion of I_6 .

7. Let $I_7 = \{Lp\}$. Using similar techniques as in 2 and in 6, one easily shows that $E(p)$ is a **T**-expansion of I_7 but not a **5**-expansion of I_7 .

Theories given in the example indicate that the only generally true inclusion relationships between the classes of expansions shown in Figure 2 are those indicated by arrows.

Theorems 3.2, 3.4 and 3.5 imply algorithms for computing \mathcal{S} -expansions of finite theories for a wide class of logics \mathcal{S} . The algorithm we give below can be used for each logic \mathcal{S} to which Theorem 3.2(b) applies and for which there exists a decision procedure for the membership problem: given a finite $I \subseteq \mathcal{L}_L$ and $\varphi \in \mathcal{L}$, does $I \vdash_S \varphi$ hold? In particular, the algorithm applies to the logics **N**, **K**, **T** and **S4**.

Algorithm:

```

compute  $I^L$ ;
for each  $\Phi \subseteq I^L$  do
     $\Psi := I^L \setminus \Phi$ ;
    if  $\Phi$  is  $\mathcal{S}$ -admissible then
        compute a finite set  $A$  such that
         $Cn(A) = Cn(I \cup \neg\Phi \cup \Psi \cup \{\psi : L\psi \in \Psi\}) \cap \mathcal{L}$ ;
        output  $E(A)$  as an  $\mathcal{S}$ -expansion of  $I$ 
    fi
  rof

```

To check \mathcal{S} -admissibility of Φ we need a decision procedure for propositional calculus to check admissibility of Φ (many such procedures are available), and a decision procedure for logic \mathcal{S} , to check whether $I \cup \neg\Phi \vdash_S \psi$, for each ψ such that $L\psi \in \Psi$. Since all the theories involved (Φ , Ψ , $I \cup \neg\Phi \cup \Psi \cup \{\psi : L\psi \in \Psi\}$ and $I \cup \neg\Phi$) are finite, verifying \mathcal{S} -admissibility can be done in a finite number of steps. For the same reason, computing the set A can be executed in finite time.

Only small changes are needed in the algorithm above to produce a method for computing all **K45**- or **W5**-expansions. One simply has to replace checking \mathcal{S} -admissibility of Φ by checking propositional admissibility of Φ (for **K45**-expansions) and strict admissibility of Φ for **W5**-expansions.

Another approach to computing expansions is possible. It is not directly based on the notion of admissibility. This other approach results in simpler and more efficient algorithms and will be presented in the full version of the paper.

We conclude this section with an example illustrating how our algorithm works. Suppose we want to compute all **S4**-expansions of the theory $I = \{\neg Lp \Rightarrow q, \neg Lq \Rightarrow p\}$. First, the theory I^L is computed. Clearly, $I^L = \{Lp, Lq\}$. There are four sets Φ that

have to be verified for S4-admissibility: $\Phi_1 = \emptyset$, $\Phi_2 = \{Lp\}$, $\Phi_3 = \{Lq\}$ and $\Phi_4 = \{Lp, Lq\}$. For the set Φ_1 , the condition $I \cup \neg\Phi_1 \vdash_{S4} \psi$, for all $L\psi \in I^L \setminus \Phi_1$ is not met. For instance, $I \cup \neg\Phi_1 \not\vdash_{S4} p$. Sets Φ_2 and Φ_3 are S4-admissible and generate S4-expansions $E(p)$ and $E(q)$. For Φ_4 , it is the case that $p \in Cn(I \cup \neg\Phi_4 \cup \Psi \cup \{\psi : L\psi \in \Psi\})$. Therefore Φ_4 is not even admissible.

4 CONCLUSIONS

In this paper we found that the structure of the family of modal nonmonotonic logics is much simpler than that of the family of underlying modal logics. This phenomenon is explained by the fact, first observed by McDermott [McD82], that the additional tool employed by modal nonmonotonic logics, namely “negation as failure to prove”, permits to prove nonmonotonically various (monotone) axiom schemata that are not provable monotonically. This phenomenon, although expected, has not been known until now.

An additional result is the demonstration of applicability of subnormal logics in the domain of knowledge representation. This seems to indicate certain incompatibility of the research in classical modal logic (where most effort has been devoted to normal logics) and the needs of knowledge representation.

Finally, expansions for a variety of modal nonmonotonic logics have been characterized and procedures to compute them described.

5 PROOFS

In this section, we give proofs of the new results of the paper.

Theorem 2.2 Let S be a modal logic, $5 \subseteq S \subseteq \text{KD45}$. Let $I, T \subseteq \mathcal{L}_L$. The theory T is a stable expansion of I if and only if T is an S -expansion of I .

Proof: It is easy to see that if $T \subseteq \mathcal{L}_L$ is stable and consistent, then $Cn(LT \cup \neg LT)$ contains all instances of axiom schemata K, D, 4 and 5. Consider for example a formula $\psi = \neg L\varphi \vee LL\varphi$, equivalent to an instance of 4. If $\varphi \notin T$, then $\neg L\varphi \in \neg LT$ and so $\psi \in Cn(\neg LT)$. If $\varphi \in T$, then $L\varphi \in T$ and $\psi \in Cn(LT)$. The remaining axiom schemata can be dealt with similarly.

In addition, $\neg LT \vdash_5 LT$. Indeed, let $\varphi \in T$. Then, since T is stable and consistent, $\neg L\varphi \notin T$, $\neg L\neg L\varphi \in T$, and $\neg L\neg L\varphi \vdash_5 L\varphi$.

These observations imply that if T is stable and consistent, then

$$Cn(I \cup LT \cup \neg LT) = Cn_S(I \cup \neg LT),$$

which immediately implies the assertion. \square

Theorem 2.3 Let S be a modal logic, $N \subseteq S \subseteq \text{WK}$. Let $I, T \subseteq \mathcal{L}_L$. The theory T is an N -expansion of I if and only if T is an S -expansion of I .

Proof: Let T be stable and consistent. Consider $\varphi, \psi \in \mathcal{L}_L$. If $\varphi \notin T$ or $(\varphi \Rightarrow \psi) \notin T$, then $L\varphi \wedge L(\varphi \Rightarrow \psi) \Rightarrow \neg L\neg L\psi \in Cn(\neg LT)$. Otherwise, $\psi \in T$ and, since T is stable and consistent, $L\psi \in T$ and $\neg L\psi \notin T$. Thus, again $L\varphi \wedge L(\varphi \Rightarrow \psi) \Rightarrow \neg L\neg L\psi \in Cn(\neg LT)$. Consequently, $Cn_{\text{WK}}(I \cup \neg LT) \subseteq Cn_N(I \cup \neg LT)$. The converse inclusion is evident. Hence $Cn_{\text{WK}}(I \cup \neg LT) = Cn_N(I \cup \neg LT)$, and the result follows. \square

Theorem 2.4 Let S be a modal logic, $\text{W5} \subseteq S \subseteq \text{D4W5}$. Let $I, T \subseteq \mathcal{L}_L$. Then, the theory T is a strict expansion of I if and only if T is an S -expansion of I .

Proof: The proof is almost identical to that of Proposition 2.2. We show that if $T \subseteq \mathcal{L}_L$ is stable and consistent, then $Cn(\{\varphi \Rightarrow L\varphi : \varphi \in T\} \cup \neg LT)$ contains all instances of axiom schemata D, 4 and W5, and that $\neg LT \vdash_{\text{W5}} \{\varphi \Rightarrow L\varphi : \varphi \in T\}$. These observations imply that if T is stable and consistent, then

$$Cn(I \cup \{\varphi \Rightarrow L\varphi : \varphi \in T\} \cup \neg LT) = Cn_S(I \cup \neg LT),$$

which immediately implies the assertion. \square

Now we will prove our results on stratification. To this end, we need two auxiliary lemmas. Let I be strongly stratified and let $I_1 \cup \dots \cup I_n$ be a stratification of I . For any propositional variable p , let $r(p) = 0$, if p does not occur in $c(\varphi)$ for any $\varphi \in I$. If p occurs in $c(\varphi)$ for $\varphi \in I_i$, then put $r(p) = i$. By the definition of strong stratifiability, such i is unique. By $r(\varphi)$ we denote $\max\{r(p) : p \text{ occurs in } \varphi\}$. By $m(\varphi)$ we denote the maximal depth of nesting of L in φ .

Lemma 5.1 Let I be strongly stratified, Φ propositionally admissible for I and let $\Psi = I^L \setminus \Phi$. Consider $L\psi \in \Psi$ such that $r(\psi) = r$ and $m(L\psi) = m$. Then

$$\begin{aligned} & I \cup \neg\Phi \cup \\ & \{L\eta \in \Psi : r(\eta) < r \text{ or } (r(\eta) = r, m(L\eta) < m)\} \vdash \psi. \end{aligned}$$

Proof: Let V be any propositional valuation such that $V(I) = 1$, $V(\Phi) = 0$ and $V(L\eta) = 1$ for each $L\eta \in \Psi$ such that $r(\eta) < r$ or $(r(\eta) = r \text{ and } m(L\eta) < m)$.

Since $\{c(\varphi) : \varphi \in I\}$ is consistent, there is a valuation W such that $W(c(\varphi)) = 1$ for each $\varphi \in I$. Define a valuation U as follows. For a propositional variable p , if $r(p) \leq r$ then put $U(p) = V(p)$. Otherwise, put $U(p) = W(p)$. For a modal atom $L\varphi$, put $U(L\varphi) = 0$, if $L\varphi \in \Phi$, and $U(L\varphi) = 1$, otherwise.

Observe that if $\varphi \in I_j$, $j > r$, then $U(c(\varphi)) = W(c(\varphi)) = 1$. Thus, $U(\varphi) = 1$. Consider $\varphi \in I_j$ with $j \leq r$. For each propositional variable p occurring in φ , $r(p) \leq r$. Hence, $U(p) = V(p)$. Let $L\alpha$ be a modal atom occurring in φ . If $L\alpha \in \Phi$, then $U(L\alpha) = 0 = V(L\alpha)$. If $L\alpha \in \Psi$ then, since $L\alpha$ occurs

in the formula $\varphi \in I_j$, $r(\varphi) < j \leq r$. Thus, $U(L\alpha) = 1 = V(L\alpha)$. Consequently, $U(\varphi) = V(\varphi) = 1$. Thus, $U(I) = 1$. By the definition of U , $U(\neg\Phi \cup \Psi) = 1$. Since Φ is propositionally admissible for I , $U(\psi) = 1$.

Now, let p be a propositional atom occurring in ψ . Then, $r(p) \leq r$ and $U(p) = V(p)$. Consider a modal atom $L\alpha$ occurring in ψ . If $L\alpha \in \Psi$, then $r(\alpha) \leq r$ and $m(L\alpha) < m$. Thus, $U(L\alpha) = V(L\alpha)$. Since U and V agree on all modal atoms $L\alpha \in \Phi$, it follows that U and V agree on all atoms occurring in ψ . Consequently, $V(\psi) = 1$. This completes the proof of the lemma. \square

Lemma 5.2 *Let I be strongly stratified and let $I_1 \cup \dots \cup I_n$ be a stratification of I . Let Φ be $\mathbf{S4}$ -admissible for I and define $\Psi = I^L \setminus \Phi$. Let $L\psi \in \Psi$ and $r(L\psi) = r$. Then*

$$I \cup \{\neg L\varphi : L\varphi \in \Phi, r(\varphi) < r\} \vdash_{\mathbf{S4}} \psi.$$

Proof: Assume that

$$I \cup \{\neg L\varphi : L\varphi \in \Phi, r(\varphi) < r\} \not\vdash_{\mathbf{S4}} \psi.$$

Then for some $\mathbf{S4}$ -Kripke model $\mathcal{N} = \langle N, Q, W \rangle$, $\mathcal{N} \models I \cup \{\neg L\varphi : L\varphi \in \Phi, r(\varphi) < r\}$, but for some $a \in N$, $\mathcal{N}, a \not\models \psi$. (Let us recall that in a Kripke model $\langle N, Q, W \rangle$, N stands for a nonempty set (of worlds), $Q \subseteq N \times N$ is an accessibility relation and for each $b \in N$, $W(b)$ is the set of all propositional variables true in the world b .)

We will construct another $\mathbf{S4}$ -Kripke model \mathcal{K} such that $\mathcal{K} \models I \cup \neg\Phi$ and $\mathcal{K}, a \not\models \psi$. This will contradict the assumption that Φ is $\mathbf{S4}$ -admissible for I , and will prove the assertion of the lemma.

First, observe that since Φ is $\mathbf{S4}$ -admissible for I , $T = T_{I, \Phi}$ is a consistent, stable theory containing $I \cup \neg\Phi \cup \Psi$. Thus, there exists a $\mathbf{S4}$ -Kripke model $\mathcal{M} = (M, R, V)$ such that $\mathcal{M} \models T$.

Next, since I is strongly stratified, there is a valuation U such that for every $\varphi \in I$, $U(c(\varphi)) = 1$. We use U to modify the valuation W of N as follows: for each $b \in N$ put

$$W'(b) = \{p : (r(p) < r \text{ and } p \in W(b)) \text{ or } (r(p) \geq r \text{ and } U(p) = 1)\}.$$

Now we define \mathcal{K} to be the concatenation of (N, Q, W') and (M, R, V) , that is

$$\mathcal{K} = (N \cup M, Q \cup (N \times M) \cup R, W' \cup V).$$

Clearly, \mathcal{K} is an $\mathbf{S4}$ -Kripke model.

By an *I-formula* we mean a formula constructed from elements of I^L and propositional variables by means of propositional connectives. We will show now by the induction on the complexity of a formula that for each *I-formula* φ such that $r(\varphi) < r$, and for each $b \in N$,

$$\mathcal{N}, b \models \varphi \text{ if and only if } \mathcal{K}, b \models \varphi. \quad (4)$$

In the case when φ is a propositional atom p with $r(p) < r$, (4) follows easily from the fact that for each world $b \in N$, $p \in W(b)$ if and only if $p \in W'(b)$. The cases when a formula φ is of the form $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$ or $\varphi_1 \wedge \varphi_2$ are easy and their discussion is omitted. Let us consider now the last case, when φ is of the form $L\gamma$. Consider $b \in N$ and assume that $\mathcal{N}, b \models L\gamma$. Then, $\mathcal{N}, c \models \gamma$ for each c such that $(b, c) \in Q$. By the induction hypothesis, $\mathcal{K}, c \models \gamma$ for each $c \in N$ such that $(b, c) \in Q$.

Since $r(L\gamma) < r$ and $\mathcal{N}, b \models L\gamma$, it follows that $L\gamma \notin \Phi$. Thus, since $\varphi = L\gamma$ is an *I-formula*, $L\gamma \in \Psi$. Consequently, since T is consistent, $\gamma \in T$. Thus, for each $c \in M$, $\mathcal{M}, c \models \gamma$. Consequently, $\mathcal{K}, b \models L\gamma$, as required. The converse implication in (4) is evident.

The property (4) implies that $\mathcal{K}, a \not\models \psi$. Let $\varphi \in I$. Consider $b \in M$. Then, $\mathcal{K}, b \models \varphi$ because $\mathcal{M}, b \models \varphi$. Assume now that $b \in N$. If $\varphi \in I_j$, $j < r$, then $\mathcal{K}, b \models \varphi$, by (4). If $\varphi \in I_j$, where $j \geq r$, then for every propositional atom p of $c(\varphi)$, $r(p) \geq r$ and, consequently, $p \in W'(b)$ if and only if $U(p) = 1$. Hence, $\mathcal{K}, b \models c(\varphi)$ which, in turn, implies that $\mathcal{K}, b \models \varphi$. Consequently, $\mathcal{K} \models I$.

Consider now $L\gamma \in \Phi$. Then $\mathcal{M} \models \neg L\gamma$. This means that for each $b \in M$, $\mathcal{K}, b \models \neg L\gamma$. In addition, it follows that for some $b_0 \in M$, $\mathcal{K}, b_0 \not\models \gamma$. Thus, by the definition of the accessibility relation of \mathcal{K} , $\mathcal{K}, b \models \neg L\gamma$, for each $b \in N$. Consequently, $\mathcal{K} \models \neg\Phi$. \square

Now, we are ready to prove Theorem 2.7.

Theorem 2.7 *Let $I \subseteq \mathcal{L}$ be strongly stratified. For each logic \mathcal{S} such that*

- (a) $N \subseteq \mathcal{S}$, and
 - (b) $\mathcal{S} \subseteq \mathbf{KD45}$ or $\mathcal{S} \subseteq \mathbf{S4}$,
- the theory I has exactly one \mathcal{S} -expansion and for all these logics \mathcal{S} , \mathcal{S} -expansions of I coincide.*

Proof: Assume that I is strongly stratified and Φ is propositionally admissible for I . Define $\Psi = I^L \setminus \Phi$ and let $L\psi \in \Psi$. Then, for each $L\psi \in \Psi$, where $\Psi = I^L \setminus \Phi$, we have $I \cup \neg\Phi \vdash_N \psi$. Indeed, this claim follows easily by induction on (r, m) , where $r = r(\psi)$ and $m = m(L\psi)$ — both the basis of the induction and the induction step follow from Lemma 5.1. In other words, for a strongly stratified theory I , if Φ is propositionally admissible for I then Φ is N -admissible for I . Consequently, each $\mathbf{KD45}$ -expansion of I is an N -expansion of I . The converse implication always holds thus, the classes of N -expansions and $\mathbf{KD45}$ -expansions coincide for strongly stratified theories. Since, by the result of [MT90a], a strongly stratified theory I has a unique $\mathbf{KD45}$ -expansion, it is also the unique N -expansion of I .

Next we show that if T_1 and T_2 are $\mathbf{S4}$ -expansions of a strongly stratified theory I , then $T_1 = T_2$. To this end we proceed as follows. Let T_1 and T_2 be determined by sets Φ_1 and Φ_2 that are $\mathbf{S4}$ -admissible for I . Put $\Psi_1 =$

$I^L \setminus \Phi_i$, $i = 1, 2$. Let r be the smallest integer such that for some $L\psi$ with $r(L\psi) = r$, $L\psi \in (\Phi_1 \setminus \Phi_2) \cup (\Phi_2 \setminus \Phi_1)$. Without loss of generality, suppose that $L\psi \in \Phi_1 \setminus \Phi_2$. Then, $L\psi \in \Psi_2$ and, by Lemma 5.2,

$$I \cup \{\neg L\varphi : L\varphi \in \Phi_2, r(\varphi) < r\} \vdash_{S4} \psi.$$

By the choice of r ,

$$\begin{aligned} \{\neg L\varphi : L\varphi \in \Phi_2, r(\varphi) < r\} = \\ \{\neg L\varphi : L\varphi \in \Phi_1, r(\varphi) < r\}. \end{aligned}$$

Thus, we obtain that $I \cup \neg\Phi_1 \vdash_{S4} L\psi$. This is a contradiction with the consistency of $I \cup \neg\Phi_1$. Thus, $\Phi_1 = \Phi_2$ and $T_1 = T_2$.

In other words, a strongly stratified theory has at most one $S4$ -expansion. Since each N -expansion is an $S4$ -expansion, that statement of the theorem follows from our previous remarks and from Proposition 2.1. \square

Theorem 3.3 (a) *Let S be any of N , K , T , $S4$, $K45$, and let Φ be S -admissible for I . Then $T_{I,\Phi}$ is the unique S -expansion of I containing $I \cup \neg\Phi$.*

(b) *Let S be any of N , K , T , $S4$. Then, T is an S -expansion of I if and only if $T = T_{I,\Phi}$, for some S -admissible Φ .*

Proof: Part (a) follows from Theorem 3.2(a). The proof of part (b), for all logics except for N , is given in Shvarts [Shv90].

To prove part (b) for logic N , one has to show that if T is an N -expansion of I , then $T = T_{I,\Phi}$, for some N -admissible set Φ . To this end, observe that T is also a K -expansion of I . Define $\Psi = T \cap I^L$ and $\Phi = I^L \setminus \Psi$. By (b) for the logic K , it follows that Φ is K -admissible for I . To complete the proof, it remains to show that Φ is N -admissible for I . We will prove a slightly stronger fact that if $I \cup \neg LT \vdash_N \gamma$, where $\gamma = \psi$ or $\gamma = L\psi$, for $L\psi \in \Psi$, then $I \cup \neg\Phi \vdash_N \gamma$.

We will proceed by the induction on the length of the proof of γ in the logic N and from $I \cup \neg LT$. If the length of the proof is 1, then γ is a tautology or $\gamma \in I \cup \neg LT$. The former case is obvious. In the latter, since $\{\gamma\}^L \subseteq I^L$, it follows that $\gamma \in I \cup \neg\Phi$.

Consider now a proof P of γ (in N and from $I \cup \neg LT$) of length $n > 1$. If γ is derived by means of the necessitation rule, then $\gamma = L\alpha$ and α has a shorter proof than γ . Moreover, by the definition of γ , $L\alpha \in \Psi$ or $LL\alpha \in \Psi$. In this latter case, since T is consistent and stable, it follows that $L\alpha \in \Psi$. Thus, in both cases the induction hypothesis applies to α and $I \cup \neg\Phi \vdash_N \alpha$. Consequently, $I \cup \neg\Phi \vdash_N L\alpha$.

Otherwise, if γ is not derived by an application of necessitation, it follows that $I \cup \neg LT \cup X \vdash \gamma$, where X is the set of all modal atoms $L\alpha$ that appear in the proof P and were derived by necessitation. Since $\{\gamma\}^L \subseteq I^L$, it follows that $I \cup \neg\Phi \cup (X \cap I^L) \vdash \gamma$. Clearly, for each atom $L\alpha \in X$, $L\alpha \in T$. Thus, $X \cap I^L \subseteq \Psi$. Since each α such that $L\alpha \in X$ has a shorter proof than γ , by

the induction hypothesis, $I \cup \neg\Phi \vdash_N \alpha$, for $L\alpha \in X$. Consequently, $I \cup \neg\Phi \vdash_N \gamma$. \square

Theorem 3.5 *If T is consistent, then T is a $W5$ -expansion of I if and only if $T = T_{I,\Phi}$ for some Φ being strictly admissible for I .*

Proof: Consider a set Φ strictly admissible for I . Define $I' = I \cup \{\varphi \Rightarrow L\varphi : L\varphi \in \Psi\}$. Then, for each $L\psi \in \Psi$,

$$I' \cup \neg\Phi \vdash \psi.$$

This implies that $I' \cup \neg\Phi \vdash_{W5} \psi$. Consequently, Φ is $W5$ -admissible for I' . Thus, by Theorem 3.1, $T = T_{I',\Phi}$ is a $W5$ -expansion of I' . By Proposition 2.4

$$\begin{aligned} T = Cn(I \cup \{\varphi \Rightarrow L\varphi : L\varphi \in \Psi\}) \\ \cup \{\varphi \Rightarrow L\varphi : \varphi \in T\} \cup \neg LT. \end{aligned}$$

But $\Psi \subseteq T_{I',\Phi} = T$. Thus, since T is consistent,

$$\{\varphi \Rightarrow L\varphi : L\varphi \in \Psi\} \subseteq \{\varphi \Rightarrow L\varphi : \varphi \in T\}.$$

Consequently,

$$T = Cn(I \cup \{\varphi \Rightarrow L\varphi : \varphi \in T\} \cup \neg LT),$$

that is, T is a $W5$ -expansion of I . It remains to show that $T_{I',\Phi} = T_{I,\Phi}$, but this follows immediately from the definition of $T_{I,\Phi}$ and the equality:

$$\begin{aligned} Cn(I \cup \neg\Phi \cup \Psi \cup \{\varphi : L\varphi \in \Psi\}) = \\ Cn(I \cup \{\varphi \Rightarrow L\varphi : L\varphi \in \Psi\} \cup \neg\Phi \cup \Psi \cup \{\varphi : L\varphi \in \Psi\}). \end{aligned}$$

To prove the “only if” part, we proceed as follows. Let T be a $W5$ -expansion of I . Then T is a $K45$ -expansion of I . Define $\Psi = T \cap I^L$ and $\Phi = I^L \setminus \Psi$. Then Φ is propositionally admissible for I and $T = T_{I,\Phi}$. It remains to prove that Φ is strictly admissible for I . Let $L\psi \in \Psi$. Then, $\psi \in T$ and, since T is a strict expansion, we have

$$I \cup \{\neg L\varphi : \varphi \notin T\} \cup \{\varphi \Rightarrow L\varphi : \varphi \in T\} \vdash \psi. \quad (5)$$

In order to show that Φ is strictly admissible, it is sufficient to prove that

$$I \cup \{\neg L\varphi : L\varphi \in I^L \setminus T\} \cup \{\varphi \Rightarrow L\varphi : L\varphi \in \Psi\} \vdash \psi. \quad (6)$$

Consider any propositional valuation V evaluating the premises of (6) as 1. Define a valuation W by

1. $W(L\varphi) = 0$, for $L\varphi \notin T \cup I^L$,
2. $W(L\varphi) = 1$, for $L\varphi \in T \setminus I^L$,
3. $W(\varphi) = V(\varphi)$, for all other atoms.

Since T is stable, $\varphi \in T$ if and only if $L\varphi \in T$. Thus, all the premises in (5) are evaluated as 1 by W . Hence $W(\psi) = 1$. Since $\{\psi\}^L \subseteq I^L$, $W(\psi) = V(\psi)$. Thus, for each valuation V satisfying the premises of (6), $V(\psi) = 1$. Consequently, (6) holds. Hence, Φ is strictly admissible for I . \square

Acknowledgements

We are very grateful to Vladimir Lifschitz for suggesting to us and encouraging our cooperation on this paper. First and third authors gratefully acknowledge the support of the Army Research Office under grant DAAL03-89-K-0124, and of National Science Foundation and the Commonwealth of Kentucky EPSCoR program under grant RII 8610671.

References

- [ABW87] K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–142, Los Altos, CA, 1987. Morgan Kaufmann.
- [Che80] B.F. Chellas. *Modal logic, and introduction*. Cambridge University Press, 1980.
- [Doy79] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.
- [Gel87] M. Gelfond. On stratified autoepistemic theories. In *Proceedings of AAAI-87*, pages 207–211, Los Altos, CA., 1987. American Association for Artificial Intelligence, Morgan Kaufmann.
- [GL88] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In *Proceedings of the 5th International Symposium on Logic Programming*, pages 1070–1080, Cambridge, MA., 1988. MIT Press.
- [GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proceedings of the 7th International Symposium on Logic Programming*, Cambridge, MA., 1990. MIT Press.
- [HC84] G.E. Hughes and M.J. Cresswell. *A companion to modal logic*. Methuen and Co. Ltd., London, 1984.
- [Kam90] M. Kaminski. Embedding a default system into nonmonotonic logic. *Fundamenta Informaticae*, 1990. To appear.
- [Kon88] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35:343–382, 1988.
- [Kon89] K. Konolige. On the relation between circumscription and autoepistemic logic. In *Proceedings of IJCAI-89*, pages 1213–1218, San Mateo, CA., 1989. Morgan Kaufmann.
- [LS90] F. Lin and Y. Shoham. Epistemic semantics for fixed-points non-monotonic logics. In *Proceedings of TARK-90*, pages 111–120, San Mateo, CA., 1990. Morgan Kaufmann.
- [Mar89] W. Marek. Stable theories in autoepistemic logic. *Fundamenta Informaticae*, 12:243–254, 1989.
- [McD82] D. McDermott. Nonmonotonic logic ii: Nonmonotonic modal theories. *Journal of the ACM*, 29:33–57, 1982.
- [MD80] D. McDermott and J. Doyle. Nonmonotonic logic i. *Artificial Intelligence*, 13:41–72, 1980.
- [Moo85] R.C. Moore. Semantical considerations on non-monotonic logic. *Artificial Intelligence*, 25:75–94, 1985.
- [MT89a] W. Marek and M. Truszczyński. Relating autoepistemic and default logics. In *Principles of Knowledge Representation and Reasoning*, pages 276–288, San Mateo, CA., 1989. Morgan Kaufmann.
- [MT89b] W. Marek and M. Truszczyński. Stable semantics for logic programs and default theories. In *Proceedings of the North American Conference on Logic Programming*, pages 243–256, Cambridge, MA., 1989. MIT Press.
- [MT90a] W. Marek and M. Truszczyński. Autoepistemic logic. *Journal of the ACM*, 1990. To appear.
- [MT90b] W. Marek and M. Truszczyński. Modal logic for default reasoning. *Annals of Mathematics and Artificial Intelligence*, 1:275 – 302, 1990.
- [Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
- [Sho87] Y. Shoham. Nonmonotonic logics: meaning and utility. In *Proceedings of IJCAI-87*, San Mateo, CA., 1987. Morgan Kaufmann.
- [Shv88] G.F. Shvarts. Stable expansions of autoepistemic theories. In *Proceedings of the All-union Conference on Artificial Intelligence*, pages 483–487, 1988. In Russian.
- [Shv90] G.F. Shvarts. Autoepistemic modal logics. In R. Parikh, editor, *Proceedings of TARK 1990*, pages 97–109, San Mateo, CA., 1990. Morgan Kaufmann.
- [Sta80] R.C. Stalnaker. A note on nonmonotonic modal logic. Unpublished manuscript, 1980.
- [Tru90] M. Truszczyński. Modal nonmonotonic logic with restricted application of the negation as failure to prove rule. To appear in: *Fundamenta Informaticae*, 1990.

Universal Attachment: An Integration Method for Logic Hybrids

Karen L. Myers
 Department of Computer Science
 Stanford University
 Stanford, CA 94305
 myers@cs.stanford.edu

Abstract

Hybrid knowledge representation and reasoning frameworks provide a powerful basis for constructing intelligent systems. *Universal attachment* is a domain-independent mechanism for integrating arbitrary types of representation and reasoning methods into hybrid frameworks that contain a logical component. Although based on the same principles as previous attachment methods, universal attachment supports a much broader range of connections between logical and other forms of representation and reasoning. Comparison with theory resolution, the leading non-attachment approach to integration, illustrates several practical advantages of universal attachment.

1 INTRODUCTION

Hybrid knowledge representation frameworks have proven to be valuable tools for the construction of nontrivial AI systems. Such frameworks incorporate multiple representation and reasoning methods into a single package, allowing different methods to be used as appropriate for storing and manipulating information. Hybrids that include a logical representation component comprise a particularly important class of frameworks. These *logic hybrids* provide the benefits of logical representations (most notably declarativism and generality) while allowing other forms of representation and reasoning to be used as required for increased efficiency and naturalness. Several systems have been constructed that attest to the utility of logic hybrids, including KRYPTON [Brachman *et al.*, 1983], KL-TWO [Vilain, 1985], CAKE [Rich, 1985], LOGIN [Ait-Kaci and Nasr, 1986] and Hyperproof [Barwise and Etchemendy, 1990].

At the core of any hybrid representation framework is the integration method used to connect the various representation components. There are many charac-

teristics that an integration method should possess; we cite four properties here. First of all, the method should support a broad range of connections between components. Second, since one of the primary motivations for using hybrid systems is increased efficiency, the integration method should be inexpensive to use. In particular, the overhead for moving between components should be minimal. Third, the integration method should promote *modular* connections. By modular, we mean that an individual component should be constructable independently of all other components. Thus an integration method should both treat reasoning methods as black boxes and not impose any predefined structural or naming conventions on representations of information. Finally, we would expect an integration method to conform to appropriate conditions of soundness and completeness for inference in hybrid systems.

These four criteria apply to integration methods for all hybrid frameworks; however, our interest lies with integration methods for logic hybrids, specifically those in which logical representation methods play a central role. With this class of hybrids, the integration problem can be reduced to that of connecting different representation and reasoning methods to a logical language.

Previous integration techniques for logic hybrids (specifically, *procedural attachment*, *semantic attachment* and *theory resolution*) fail to satisfy the criteria stated above. In this paper we present a new integration technique, *universal attachment*, that does. Universal attachment is similar to previous attachment approaches in that data and programs from different components are ‘attached’ to expressions in the language of the logical component. When expressions with attachments are encountered during theorem proving, the attached data and programs are used to directly evaluate those expressions, rather than employing further deduction. In this paper, we show that universal attachment is a more powerful integration method than either procedural attachment or semantic attachment, the previous attachment approaches. As

well, universal attachment supports connections having greater modularity. A comparison with theory resolution shows that universal attachment provides both broader applicability and increased modularity while being easier to control.

In Section 2 we define universal attachment. Section 3 presents a specialized inference rule of universal attachment tailored to clausal resolution systems and establishes the soundness of that rule. Section 4 surveys previous integration methods for logic hybrids and compares them with universal attachment. In Section 5, we discuss tools that manipulate universal attachments in order to assist a user of a hybrid system.

Experimental results described in this paper were obtained using a universal attachment framework implemented on top of the KADS theorem proving system [Stickel, 1988].

2 UNIVERSAL ATTACHMENT

In this section, we define *universal attachment* (UA). The descriptor *universal* derives from the generality of the method: universal attachments can be defined for any expression (term or formula) in a first-order language. This capability contrasts with previous attachment methods which only permit attachments for instances of functions and relations.

2.1 A DEFINITION OF UA

As described above, universal attachments draw connections from expressions in a first-order language \mathcal{L} to programs and data structures in alternative languages. Characterization of an individual universal attachment requires descriptions of both the class of expressions to which the attachment applies (referred to as the *domain* of the attachment) and a method for evaluating the attachment. We provide such a characterization here. Let $\phi[\theta_1, \dots, \theta_k]$ represent an expression $\phi \in \mathcal{L}$ that contains one or more occurrences of the subexpressions $\theta_1, \dots, \theta_k$.

The domain of a universal attachment is determined by two parameters, the *attachment pattern* and the *filter set*. The attachment pattern, denoted by $\phi[x_1, \dots, x_n]$, is a schematic representation of a class of expressions in \mathcal{L} , defined in terms of the variables x_1, \dots, x_n .¹ The x_i , referred to as the *distinguished variables* for the pattern, are assumed to encompass all unbound variables in ϕ . The filter set, denoted by F , is a collection of constraints on instantiations of distinguished variables in the attachment pattern.

¹Throughout this document, italicized text denotes symbols from \mathcal{L} (e.g. *subsumes*(A, B)), typewriter font indicates LISP objects (e.g. pathp) and sans serif font indicates metalevel constructs (e.g. EXPR). Constants in \mathcal{L} are capitalized while variables appear in lower case.

The domain of the universal attachment is defined to be the set of instantiations $\phi[t_1, \dots, t_n]$ of the attachment pattern where the t_i are terms in \mathcal{L} that satisfy the constraints in F (written as $[t_1, \dots, t_n]$ satisfies F). The choice of a particular language for expressing constraints is independent of the notion of universal attachment, thus different filter languages can be employed as required for the application at hand. A more detailed discussion of filter languages is presented in Section 2.2.

The evaluation process for a universal attachment makes use of a program p , referred to as the *attached program*, as well as an *attachment map* and a *detachment map*, denoted by A and D , respectively. These maps provide an important service in that they translate between terms in \mathcal{L} and attached structures in some alternative language used to represent the objects denoted by those terms. In particular, the *attachment map* A translates the instantiations of the distinguished variables onto the appropriate set of parameters for the attached program p . The *detachment map* translates the result of evaluating p on the attached data onto the appropriate expression in the logical language. This last result is referred to as the *value* of the universal attachment.

In order to characterize the class of programs and data to which universal attachments are made, we introduce the concept of a *computation system*. A computation system corresponds to an individual representation component within a hybrid system. Computation systems consist of three parts: *programs*, *data structures* and an *evaluation function*. Programs correspond to inference rules or evaluation methods and are applied to data structures; the number of inputs for a particular program is called the *arity* of that program. The evaluation function maps a program and data structures onto either a resultant data structure or \perp . The result \perp indicates that the program is undefined for the given set of data structures.

Definition 2.1 (Computation System) A computation system is a triple $(\mathcal{P}, \mathcal{D}, E)$ where \mathcal{P} is a set of programs, \mathcal{D} is a set of data structures, and E is an evaluation function $E : \mathcal{P} \times \mathcal{D}^* \rightarrow \mathcal{D} \cup \{\perp\}$ that maps a program $p \in \mathcal{P}$ of arity m and data structures $d_1, \dots, d_m \in \mathcal{D}$ onto either some $d \in \mathcal{D}$ or \perp .

In a computation system containing LISP programs and data structures, E would be the LISP function eval. For simplicity, we assume the use of a LISP-based computation system throughout this document.

A computation system describes one cornerstone of the framework for which universal attachments can be made. The language \mathcal{L} and the filter language constitute the others.

Definition 2.2 (UA Framework) A universal attachment framework is a five-tuple $(\mathcal{L}, F, \mathcal{P}, \mathcal{D}, E)$

where \mathcal{L} is a first-order language, \mathcal{F} a filter language, and $(\mathcal{P}, \mathcal{D}, \mathcal{E})$ a computation system.

We now formalize the intuitive notion of universal attachment presented above. Let $\text{TERMS}(\mathcal{L})$ represent the terms of \mathcal{L} and let $\text{EXPR}(\mathcal{L})$ represent the expressions of \mathcal{L} (i.e. the terms and formulas of \mathcal{L}).

Definition 2.3 (Universal Attachment)

For a UA framework $(\mathcal{L}, \mathcal{F}, \mathcal{P}, \mathcal{D}, \mathcal{E})$, a universal attachment is a five-tuple denote by

$$\langle \phi[x_1, \dots, x_n], F \rangle \rightarrow \langle p, A, D \rangle \quad (1)$$

with components

- $\phi[x_1, \dots, x_n]$, the attachment pattern, a schematic representation of a subset of $\text{EXPR}(\mathcal{L})$,
- F , the filter set, a set of constraints defined for x_1, \dots, x_n using language \mathcal{F} .
- p , the attached program, an element of \mathcal{P} having arity m ,
- A , the attachment map, a function $A : \text{TERMS}(\mathcal{L})^n \rightarrow \mathcal{D}^m$,
- D , the detachment map, a function $D : \mathcal{D} \rightarrow \text{EXPR}(\mathcal{L})$.

We can also formally define the *domain* and *value* of a universal attachment. In doing so, we adopt the notation \hat{t} to represent an n -ary vector of terms.

Definition 2.4 (UA Domain) For a universal attachment δ given as $\langle \phi[x_1, \dots, x_n], F \rangle \rightarrow \langle p, A, D \rangle$, the domain of δ , denoted by $\text{DOMAIN}(\delta)$, is defined by

$$\text{DOMAIN}(\delta) = \bigcup_{\hat{t} \in \text{TERMS}(\mathcal{L})^n} \{\phi[\hat{t}] \mid \hat{t} \text{ satisfies } F\}.$$

Definition 2.5 (UA Value) For a universal attachment δ given as $\langle \phi[x_1, \dots, x_n], F \rangle \rightarrow \langle p, A, D \rangle$, the value of δ applied to $\phi[t_1, \dots, t_n] \in \text{DOMAIN}(\delta)$, is represented as $\text{ATTACH}(\delta, \phi[t_1, \dots, t_n])$ and defined by

$$\text{ATTACH}(\delta, \phi[t_1, \dots, t_n]) = D(E(p, A([t_1, \dots, t_n]))).$$

We note that the value of a universal attachment δ may be undefined for some expressions in its domain since E is not required to be a total function.

The interpretation of the universal attachment (1) is that any occurrence within a formula of \mathcal{L} of an expression in its domain can be replaced by the value of the attachment for that expression. More formally, we have the following inference rule for universal attachment.

Definition 2.6 (UA Inference Rule) For a universal attachment δ and formula Q , if $\phi \in \text{DOMAIN}(\delta)$ and $\theta = \text{ATTACH}(\delta, \phi)$ then

$$\frac{Q[\phi]}{Q[\theta]} . \quad (2)$$

Example 2.7 (Inheritance) Consider the task of reasoning about an inheritance hierarchy. We could formulate logical axioms describing the relevant subsumption relationships using a binary predicate $\text{subsumes}(x, y)$ whose intended interpretation is that class x subsumes class y . We would require sentences of the form $\text{subsumes}(A, B)$, $\text{subsumes}(A, C)$, etc. as well as axioms describing general properties of subsumption (e.g. subsumes is transitive).

Alternatively, we could employ a universal attachment for subsumes that makes use of a directed graph representing the inheritance hierarchy and an efficient path-finding procedure. In particular, we could define a LISP list ih that represents the inheritance hierarchy described by subsumes in terms of LISP atoms a, b, c , etc. As well, we would need a LISP function $\text{pathp}(n1, n2, g)$ that returns t if there is a path from $n1$ to $n2$ in the directed graph g , and nil otherwise. Finally, we would require a metalevel function h that maps logical representations of classes onto the LISP representations of those classes (i.e. $h(A) = a$, $h(B) = b$, etc). Using these components, we could define the universal attachment:

$$\begin{aligned} \langle \text{subsumes}(x, y), \{\} \rangle &\rightarrow \langle \text{pathp}, A^{\text{path}}, D^{\text{tf}} \rangle \quad (3) \\ A^{\text{path}}([x, y]) &= (h(x), h(y), \text{ih}) . \end{aligned}$$

The domain of this universal attachment is all instances of $\text{subsumes}(x, y)$. The attachment map A^{path} translates a pair of logical terms denoting classes into the triple consisting of the LISP representation of those classes and the graph ih . The detachment map D^{tf} translates the LISP atom t to the expression *true* and *false*, where *true* and *false* are special zeroary predicates in \mathcal{L} having the intended interpretations of truth and falsity, respectively.

Example 2.8 (Quantified Attachments)

Let cyclicp be a LISP function that takes one parameter g and determines whether the directed graph g is cyclic. Given the transitivity and irreflexivity of subsumes , the condition of cyclicity for the taxonomy described above is captured by the formula

$$\exists x. \text{subsumes}(x, x) .$$

If A^{ih} is the zero-ary function with value ih , we can define the following universal attachment for the quantified formula $\exists x. \text{subsumes}(x, x)$:

$$\langle \exists x. \text{subsumes}(x, x), \{\} \rangle \rightarrow \langle \text{cyclicp}, A^{\text{ih}}, D^{\text{tf}} \rangle . \quad (4)$$

With the UA inference rule (2), it is assumed that the specific identifiers chosen to represent bound variables are irrelevant, thus $\exists y. \text{subsumes}(y, y)$ for any choice of y would be in the domain of the attachment (4).

A simple modification to Definition 2.5 makes it possible to used attached programs that are *partial* in the sense that a lack of information prevents them for always returning a satisfactory answer. This capability would be useful, for example, if instead of the program *cyclicp* in Example 2.8 we had a program *cyclicp1* that operated on a list representing a partial description of a graph and returned one of *t*, *nil* or *unknown*, depending on whether the graph was cyclic, the graph was not cyclic, or the cyclicity or noncyclicity of the graph could not be determined. To accommodate partial programs, we redefine D to be a function

$$D : \mathcal{D} \rightarrow \text{EXPR}(\mathcal{L}) \bigcup \{\text{fail}\}$$

and set the value of the universal attachment $\langle \phi[x_1, \dots, x_n], F \rangle \rightarrow \langle p, A, D \rangle$ to be

$$\text{ATTACH}(\delta, \phi[t_1, \dots, t_n]) = \phi[t_1, \dots, t_n]$$

for $D(E(p, A([t_1, \dots, t_n]))) = \text{fail}$. Universal attachments to partial programs would employ detachment maps that translate *unknown* to *fail*; these attachments reduce to the identity transformation when the attached program evaluates to *unknown*.

2.2 FILTER SETS

As described above, a filter set is a collection of constraints on instantiations of the distinguished variables in attachment patterns. These constraints must be satisfied in order for a given instantiation of the pattern to be in the domain of the universal attachment.

In order to appreciate the need for filter sets, consider the specification of a universal attachment for instances of the logical function *plus(x, y)* whose interpretation is that of integer arithmetic. Using the LISP addition routine *+* and attachment maps A^{int} and D^{int} defined to translate between terms in \mathcal{L} denoting integers and the corresponding LISP symbols for those integers, we can define an attachment

$$\langle \text{plus}(x, y), \{\} \rangle \rightarrow \langle +, A^{\text{int}}, D^{\text{int}} \rangle . \quad (5)$$

However, the domain of this universal attachment (namely *plus(x, y)* for all instantiations of x and y) is too large. If our attached program only operates on integers or A^{int} is only defined for terms denoting integers then we must exclude from the domain those expressions in which x and y are bound to terms that do not denote integers. Such exclusions require information about sorts. Furthermore, although the logical

constant *the-pope's-favourite-number* has sort integer, we would not want to attach to expressions in which this constant is bound to a distinguished variable since we don't know the specific number that this constant denotes. Thus we need the means to express syntactic constraints in addition to sort constraints.

The expressiveness of the language used to specify filters certainly impacts the power of the universal attachment framework as a whole. Because the filter language can be defined independently of the universal attachment framework, a different filter language can be utilized according to the nature of the applications being considered. We describe here a simple but powerful filter language $\mathcal{F}^*(\mathcal{L})$. The language consists of the metalevel predicates *atomic(t)*, *sort(t,s)*, and *standard-name(t,s)* (referred to as the *filters* of $\mathcal{F}^*(\mathcal{L})$), with their satisfaction defined as follows.

Definition 2.9 (Filter Satisfaction for $\mathcal{F}^*(\mathcal{L})$) A term t in \mathcal{L} satisfies a filter f in $\mathcal{F}^*(\mathcal{L})$ iff:

1. f is *atomic(t)* and t is a constant symbol in \mathcal{L} .
2. f is *sort(t,s)* and t has sort s .
3. f is *std-name(t,s)* and t is a standard name for sort s .

The filter *atomic* requires that its argument be a constant symbol in \mathcal{L} . The filter *sort* imposes sort restrictions on terms; corresponding sort information must be provided by the user of the attachment system. As an illustration, in Example 2.7 it may be that the constants in \mathcal{L} of sort class cover all classes for which we want the attachment (3) to apply. In this case we would adopt the filter set

$$F^h = \{\text{sort}(x, \text{class}), \text{atomic}(x), \text{sort}(y, \text{class}), \text{atomic}(y)\} .$$

Standard name filters provide syntactic distinctions within sorts. These filters constrain their arguments to be terms in \mathcal{L} that have been deemed *standard* for a given sort. For example, we might specify that all numeric representations of integers within our logical language are standard names for the sort integer. Standard name filters could be used to define an appropriate filter set for the attachment to *plus(x, y)* given by (5) as follows:

$$F^+ = \{\text{std-name}(x, \text{integer}), \text{std-name}(y, \text{integer})\} .$$

$\mathcal{F}^*(\mathcal{L})$ is used as the filter language in our implemented universal attachment system. Despite the simplicity of $\mathcal{F}^*(\mathcal{L})$, the expressive power that it provides has proven to be satisfactory in practice. $\mathcal{F}^*(\mathcal{L})$ has the additional benefits that little information is required to validate the satisfaction of its filters and that the validation process is inexpensive. In the current implementation, validation of the filter *atomic* is achieved

by a simple check on the definition of \mathcal{L} . The filter sort requires only conventional sort information, which is readily stored and accessed by the sort facility within KADS. A mechanism for defining sets of standard names has been added to KADS to support the filter std-name. The mechanism supports explicit enumerations of standard names, the use of characteristic functions for recognizing standard names, and recursive definitions of standard names. This last capability is useful for sorts such as path whose standard names could be defined to be the constant *nil* (denoting the null path) along with terms consisting of the function *join* applied to a first argument that is a standard name for the sort node and a second argument that is a standard name for the sort path.

Filtering could be realized within the attached procedures themselves, thus obviating the need for an explicit filtering mechanism. That is precisely how procedural attachment works: the attached procedure is designed to detect unsuitable parameters and to terminate an attachment involving such parameters. The cost of embedding the filtering process in attached procedures is a substantial loss of modularity for attached programs. Imposing filtering responsibility on attached procedures precludes the use of existing libraries of procedures, since the attached procedures would have to be tailored to the logical language being used.

3 ALTERNATIVE INFERENCE RULES

The UA inference rule (2) employs a definition of containment based on syntactic matching of one or more subexpressions. In this section, we illustrate how this notion of containment limits the applicability of universal attachments. Using an alternative definition of containment, we define a new inference rule of universal attachment for use with clausal resolution systems.

3.1 THE CONTAINMENT PROBLEM

The inference rule (2) states that a universal attachment δ can be applied to a formula $Q[\phi]$ yielding $Q[\theta]$ provided ϕ is in the domain of δ and δ evaluates to θ on ϕ . The notation $Q[\phi]$ was interpreted to mean “ Q contains the expression ϕ ”, with containment being interpreted as exact syntactic matching of a subexpression.

This purely syntactic notion of containment is not always satisfactory. Consider a universal attachment δ defined for the conjunction $P \wedge Q$. With the UA inference rule (2), δ is not applicable to the formula $Q \wedge P$ despite the fact that $P \wedge Q \equiv Q \wedge P$. Similarly, δ could not be used to simplify the expression $P \wedge R \wedge Q$. In situations where such applications of δ are desired, a stronger form of UA inference must be defined.

The definition of containment as exact subexpression matching also presents problems for resolution proof systems. Such systems rely on normalized representations of axioms in which quantification is implicit: variables are interpreted as being universally quantified while Skolem terms represent existentially quantified variables. With these normalized representations, exact subexpression matching suffices when applying attachments defined for functions and relations but problems arise with attachments defined for quantified expressions. In order to use quantified attachments with these representations, a more sophisticated notion of containment than syntactic pattern matching of subexpressions is required that accounts for implicit quantification.

3.2 UNIVERSAL ATTACHMENT AND CLAUSAL FORM

Given the appeal of resolution theorem provers, we present an inference rule of universal attachment based on a definition of containment tailored to normalized representations. We focus on clausal representations here although the ideas are equally applicable to the nonclausal case. For simplicity, we restrict attention to attachments defined for function and relation instances (referred to as *function* and *relation* attachments, respectively) and for quantified expressions whose matrices are atoms (referred to as *simple quantified attachments*). We also assume that relation and simple quantified attachments always evaluate to either *true* or *false*.

Before presenting a formal definition of the inference rule, we examine the issue of containment for simple quantified attachments. Consider a universal attachment for the quantified formula $\forall x. P(x)$. Although the normalized representation of this formula is $P(x)$ for any choice of variable x , we cannot apply our attachment to arbitrary occurrences of this expression. To see why, consider the clause $P(x) \vee Q(x)$ which is a normalized representation of the formula $\forall x. P(x) \vee Q(x)$. If the universal attachment for $\forall x. P(x)$ evaluates to *false* and we substitute this value for $P(x)$ in $P(x) \vee Q(x)$, the result simplifies to the unsound conclusion $Q(x)$. The unsoundness results because the literal $P(x)$ does not correspond to $\forall x. P(x)$ in this case since $P(x) \vee Q(x)$ represents the formula $\forall x. P(x) \vee Q(x)$ rather than the formula $\forall x. P(x) \vee \forall x. Q(x)$. On the other hand, we would expect the attachment to apply to the occurrence of $P(x)$ in $P(x) \vee R$ since this clause represents the formula $\forall x. P(x) \vee R$. More generally, to guarantee that occurrences of $P(x)$ correspond to $\forall x. P(x)$ it is necessary that the variable of quantification not appear elsewhere in the clause to which the attachment is being applied. As well, we could not apply an attachment for $\forall xy. H(x, y)$ to the literal $H(x, x)$ since this literal constrains its two arguments to be identical. Note that the

presence of the same variable in different clauses is not a problem since such variables can be renamed without affecting soundness. This is a consequence of the fact that universal quantification distributes over conjunction (*i.e.* $\forall x. P(x) \wedge Q(x) \equiv \forall x. P(x) \wedge \forall x. Q(x)$).

The attachment for $\forall x. P(x)$ should also apply to negative instances of $\forall x. P(x)$ occurring in normalized representations of formulas. These negative instances appear as expressions with Skolem terms; for example, if S_k is a Skolem term then $\neg P(S_k)$ is a normalized representation of $\neg \forall x. P(x)$. In analogy with the case for variables in positive instances, problems arise with multiple occurrences of Skolem terms. Occurrences of S_k in other literals of the clause, as with the clause $P(S_k) \vee Q(S_k)$, can be safely ignored since existential quantifiers distribute over disjunction (*i.e.* $\exists x. P(x) \vee Q(x) \equiv \exists x. P(x) \vee \exists x. Q(x)$). However, occurrences of S_k within the same literal cause problems. For example an attachment for $\exists x y. H(x, y)$ should not be applied to $H(S_k, S_k)$ since this latter literal represents $\exists x. H(x, x)$. Although it may seem that occurrences of S_k in different clauses would pose difficulties (since existential quantification does not distribute over conjunction), such occurrences can be ignored when applying the quantified attachment. To see why, consider the two values (*true* or *false*) that the attachment can assume. If the attachment for $\forall x. P(x)$ evaluates to *false* then $\neg P(S_k)$ is replaced by *true* and the resulting clause is a tautology. If the attachment for $\forall x. P(x)$ evaluates to *true* then $\neg P(S_k)$ would be replaced by *false*, a step which is intuitively sound.

Attachments defined for existential formulas are treated in dual fashion. These observations lead to a variation on the inference rule of universal attachment (2) for use with clausal resolution theorem provers. We call the result UAR (from Universal Attachment for Resolution). UAR is comprised of three different rules, corresponding to function and relation attachments, simple quantified attachments applied to positive literals and simple quantified attachments applied to negated literals. We note that the function and relation rule is a specialization of the inference rule (2).

Definition 3.1 (UAR) Let δ be a universal attachment $\langle \phi[x_1, \dots, x_n], F \rangle \rightarrow \langle p, A, D \rangle$. Define the UAR inference rules for clausal form as follows.

Function and Relation Attachment Rule If ϕ is a function or relation instance then for any literal L , clause C and terms t_1, \dots, t_n in L define the quantifier-free rule of universal attachment by

$$\frac{C \cup \{L[\phi[t_1, \dots, t_n]]\}}{C \cup \{L[\theta]\}} \quad (6)$$

where $\theta = \text{ATTACH}(\delta, \phi[t_1, \dots, t_n])$, provided $[t_1, \dots, t_n]$ satisfies F .

Simple Quantified Attachments: Positive Case If $\phi[x_1, \dots, x_n]$ is a simple quantified expression $Q_1 z_1 \dots Q_k z_k. \psi[z_1, \dots, z_k, x_1, \dots, x_n]$ for quantifiers Q_1, \dots, Q_k and relation ψ , then for any clause C and terms t_1, \dots, t_n in L define the positive quantified rule of universal attachment by

$$\frac{C \cup \{\psi[s_1, \dots, s_k, t_1, \dots, t_n]\}}{C \cup \{\theta\}} \quad (7)$$

where $\theta = \text{ATTACH}(\delta, \psi[t_1, \dots, t_n])$, provided $[t_1, \dots, t_n]$ satisfies F and for $i = 1 \dots k$:

- If Q_i is \forall then s_i is a variable that does not appear in C
- For variables s_i and s_j , if $s_i = s_j$ then $i = j$.

Simple Quantified Attachments: Negative Case

If $\phi[x_1, \dots, x_n]$ is a simple quantified expression $Q_1 z_1 \dots Q_k z_k. \psi[z_1, \dots, z_k, x_1, \dots, x_n]$ for quantifiers Q_1, \dots, Q_k and relation ψ , then for any clause C and terms t_1, \dots, t_n in L define the negative quantified rule of universal attachment by

$$\frac{C \cup \{\neg\psi[s_1, \dots, s_k, t_1, \dots, t_n]\}}{C \cup \{\neg\theta\}} \quad (8)$$

where $\text{ATTACH}(\delta, \psi[t_1, \dots, t_n]) = \theta$, provided $[t_1, \dots, t_n]$ satisfies F , and for $i = 1 \dots k$:

- If Q_i is \exists then s_i is a variable that does not appear in C .
- For variables s_i and s_j , if $s_i = s_j$ then $i = j$.

The definition of containment underlying UAR allows attachments to be applied in a manner that goes beyond the ‘substitution of equals’ traditionally associated with attachments. For example, (7) sanctions application of an attachment for $\exists x. P(x)$ to $P(a)$. When the attachment evaluates to *true* the result is a tautologous clause; when the attachment evaluates to *false* the result is the empty clause. Although these applications are intuitively sound, the expressions $\exists x. P(x)$ and $P(a)$ are not logically equivalent.²

3.3 SOUNDNESS OF UAR

As with all inference rules, we would like to ensure that resolution augmented with UAR produces only sound conclusions. Because a computation system in a universal attachment framework can employ arbitrary evaluation methods, conventional model-theoretic notions are insufficient for defining a soundness criteria for attachment inference. An appropriate definition of

²A more complex version of UAR can be readily formulated that prohibits such non-substitutive applications.

soundness must take into account the information provided by attachments. To this end we introduce the notion of an *axiom set* for a universal attachment. Let \hat{t} be an n -ary vector and let T^n be the variable-free n -tuples of $\text{TERMS}(\mathcal{L}) \times \dots \times \text{TERMS}(\mathcal{L})$.

Definition 3.2 (Axiom Set) Let δ be a universal attachment $\langle \phi[x_1, \dots, x_n], F \rangle \rightarrow \langle p, A, D \rangle$. If ϕ is a functional expression then define the axiom set for δ , represented by $\text{AXIOMS}(\delta)$, to be

$$\bigcup_{\hat{t} \in T^n} \{\phi[\hat{t}] = \theta \mid \hat{t} \text{ satisfies } F \text{ and } \text{ATTACH}(\delta, \phi[\hat{t}]) = \theta\}.$$

If ϕ is non-functional then define $\text{AXIOMS}(\delta)$ to be the union of the following two sets:

$$\bigcup_{\hat{t} \in T^n} \{\phi[\hat{t}] \mid \hat{t} \text{ satisfies } F \text{ and } \text{ATTACH}(\delta, \phi[\hat{t}]) = \text{true}\}$$

$$\bigcup_{\hat{t} \in T^n} \{\neg\phi[\hat{t}] \mid \hat{t} \text{ satisfies } F \text{ and } \text{ATTACH}(\delta, \phi[\hat{t}]) = \text{false}\}.$$

Example 3.3 Given $\{0, 1, -1, 2, \dots\}$ as the standard names for the sort integer, the axiom set for the attachment

$$\langle \text{plus}(x, y), F^+ \rangle \rightarrow \langle +, A^{\text{nat}}, D^{\text{nat}} \rangle$$

$$F^+ = \{\text{std-name}(x, \text{integer}), \text{std-name}(y, \text{integer})\}$$

is

$$\{\text{plus}(0, 0) = 0, \text{plus}(0, 1) = 1, \text{plus}(0, -1) = -1, \dots\}.$$

Example 3.4 Suppose the taxonomy described by *subsumes* in Example 2.7 consists of the root node A with descendants B and C . As well, assume that the set of constants in \mathcal{L} of sort *class* is $\{A, B, C, D\}$. The axiom set for the attachment

$$\langle \text{subsumes}(x, y), F^{\text{ih}} \rangle \rightarrow \langle \text{pathp}, A^{\text{path}}, D^{\text{tf}} \rangle$$

$$F^{\text{ih}} = \{\text{sort}(x, \text{class}), \text{atomic}(x), \text{sort}(y, \text{class}), \text{atomic}(y)\}$$

would then contain the positive atoms

$$\{\text{subsumes}(A, B), \text{subsumes}(A, C)\} \quad (9)$$

and the negated atoms $\neg\text{subsumes}(x, y)$ for x and y in $\{A, B, C, D\}$ that are consistent with the sentences in (9).

We will refer to a collection of universal attachments as a *system* of attachments and define the axiom set for a system as follows.

Definition 3.5 If Δ is a system of universal attachments, then the axiom set $\text{AXIOMS}(\Delta)$ for Δ is defined by

$$\text{AXIOMS}(\Delta) = \bigcup_{\delta \in \Delta} \text{AXIOMS}(\delta).$$

Using the concept of axiom sets, we can state an appropriate notion of soundness for proof systems containing inference rules of attachment. Let $\mathcal{E}(\mathcal{L})$ represent the *equality axioms* for \mathcal{L} (i.e. the axioms of reflexivity, commutativity, and transitivity for equality along with the axioms of substitutivity for the functions and predicates in \mathcal{L}). We say that a proof system containing inference rules of attachment is sound if every deductive consequence of that system for a given set of sentences S and attachments Δ is a logical consequence of $S \cup \text{AXIOMS}(\Delta) \cup \mathcal{E}(\mathcal{L})$. The following theorem establishes the soundness of resolution augmented with UAR.

Theorem 3.6 (Soundness for UAR) Let Δ be a system of function, relation and simple quantified attachments in \mathcal{L} . If there is a derivation of the empty clause from a clausal representation of a set S of sentences in \mathcal{L} using resolution and UAR for Δ then $S \cup \text{AXIOMS}(\Delta) \cup \mathcal{E}(\mathcal{L})$ is unsatisfiable.

Proof. We sketch the proof here; details can be found in [Myers, 1991]. If T is a theory having clausal form $C(T)$ then T is unsatisfiable if and only if $C(T)$ is unsatisfiable [Chang and Lee, 1973]. Given the soundness of resolution and paramodulation for theories with equality, it suffices to show that if there is a derivation of the empty clause from a clausal representation of S using resolution and UAR for Δ then there is a derivation from the clausal representation of $S \cup \text{AXIOMS}(\Delta) \cup \mathcal{E}(\mathcal{L})$ using resolution and paramodulation. Suppose application of an attachment $\delta \in \Delta$ to a clause D produces a non-tautologous clause D' . If δ is defined for functional expressions we can show that paramodulation can be applied to D using some equality relation in $C(\text{AXIOMS}(\delta))$ (where $C(\text{AXIOMS}(\delta))$ is a clausal form for $\text{AXIOMS}(\delta)$) to produce D' . Similarly, if δ is an attachment defined for non-functional expressions then there is a literal in $C(\text{AXIOMS}(\delta))$ that can be resolved with D to yield D' . The result immediately follows. \square

We note that the soundness of the inference rule (2) is straightforward to establish given its definition of containment as syntactic matching of subexpressions.

Theorem 3.6 guarantees that application of attachments using UAR produces results that are logical consequences of the given axioms and the axiom sets of the attachments. However, unintended inferences can still enter into an attachment system if attached programs and representations are incorrectly defined (i.e. the axiom sets do not match the intended semantics for attached programs and representations). For many applications, we believe that concern about the introduction of unintended inferences is outweighed by the increased representational capabilities and inferential efficiency that attachments can provide. In situations where traditional notions of soundness are crucial, one

could apply program verification techniques to ensure that attached programs are appropriately defined.

In addition to being sound, we would like our inference rule of attachment to satisfy an appropriate condition of completeness. Informally, we say that an inference rule of attachment is complete if inferential consequences are preserved when information is moved from axiomatic representations to attached representations, provided that appropriate attachments are supplied. More specifically, given a set of sentences S and a system of attachments Δ , completeness requires that when $S \cup \text{AXIOMS}(\Delta) \vdash \phi$ there is a derivation of ϕ from S using the original inference mechanism along with the inference rule of universal attachment applied for Δ . Completeness ensures, for instance, that replacement of axioms describing a graph by both a data structure representing that graph and attachments for extracting information from that graph should not affect the set of inferences that the overall system can make. It is shown in [Myers, 1991] that completeness holds for resolution augmented with UAR+ (an extension of UAR that handles nonsimple quantified attachments) provided that the system of universal attachments contains certain quantified attachments.

4 ALTERNATIVE INTEGRATION METHODS

We now turn to the relationship between universal attachment and alternative integration methods for logic hybrids, specifically procedural attachment, semantic attachment and theory resolution.

4.1 ATTACHMENT METHODS

As described above, attachment methods are characterized by the use of links called *attachments* from individual expressions in a logical language to evaluation mechanisms outside of the logical representation component. From a formal perspective, attachments can be characterized by the following generalization of the UA inference rule (2).

Definition 4.1 (Generic Rule of Attachment)
If there is an attachment for ϕ that evaluates to θ then for any formula Q

$$\frac{Q[\phi]}{Q[\theta]} .$$

The various attachment methods, *procedural attachment*, *semantic attachment* and *universal attachment*, differ in their approaches to specification and evaluation.

4.1.1 Procedural Attachment

The simplest form of attachment is *procedural attachment* in which attachments are specified as links from

logical function and relation names to programs. Evaluation simply consists of executing these attached programs. Metafunctions in the Boyer-Moore theorem prover [Boyer and Moore, 1981], attachments in MRS [Russell, 1985], and Green's predicate evaluation [Green, 1969] are all forms of procedural attachment. One can also view programming languages that merge relational and functional programming (*e.g.* TABLOG [Malachi *et al.*, 1986] and LOGLISP [Robinson and Sibert, 1982]) as relational languages enhanced with procedural attachment.

Procedural attachment is a simple yet powerful mechanism for incorporating certain forms of information into logical representation systems. However, its utility as a general integration tool is limited. Procedural attachment provides a very weak form of integration due to its reliance on the *symbol sharing assumption*: parameters for attached programs are assumed to be named by the symbols appearing as arguments in the expression for which the attachment is being applied. The symbol sharing assumption severely limits the attachments that can be made and causes the attachments that are possible to be highly unmodular. As was noted in Section 2.2, the lack of explicit mechanisms for filtering also restricts the modularity of procedural attachment.

4.1.2 Semantic Attachment

Semantic attachment [Weyhrauch, 1977; Weyhrauch, 1980] provides a more powerful attachment alternative to procedural attachment by virtue of its explicit attachments to data. Semantic attachments are specified from individual constant names in the logical language to data structures and from individual function and relation names in the logical language to programs. A given logical expression $\alpha(a_1, \dots, a_n)$ can be evaluated using semantic attachment only when α and each a_i have attachments. Evaluation proceeds in a manner that mirrors the compositionality of denotational semantics: the program attached to α is executed on the parameters corresponding to the attached data structures for a_1, \dots, a_n . The value returned by the program is then translated back to the logical name which is attached to that value.

Semantic and universal attachment both satisfy the criteria of effectiveness and modularity for integration methods. However, universal attachment generalizes semantic attachment in a way that provides greater expressivity. Much of the increased expressiveness derives from the richer translation facility provided by universal attachment. By its nature, semantic attachment requires a one-to-one correspondence between arguments in the logical expression for which the attachment is defined and the parameters for the attached program. This restriction precludes the use of a semantic attachment for the predicate *subsumes* to the program *pathp* in Example 2.7 since *subsumes* has ar-

ity two while `pathp` has arity three. More generally, semantic attachment cannot make use of attached programs that require explicit representations of objects that are implicit in the axiomatization.

A second source of increased expressivity for universal attachment stems from the ability to handle attachments for all logical expressions, not just instances of functions and relations. Our experimental results show that attachments defined for quantified expressions can greatly increase reasoning efficiency. Intuitively, a program attached to a quantified expression can achieve greater efficiency than theorem proving because the program can focus on an appropriately limited set of objects that are relevant to the quantification. For example, consider a ternary relation $\text{conn}(x, y, g)$ whose interpretation is that nodes x and y are connected in the graph g . For the formula $\forall x \ y. \text{conn}(x, y, g)$ whose interpretation is that graph g is a clique, we can define an attachment to an appropriately defined program `cliquep(g)`. While the program `cliquep` need only consider the nodes contained within a particular graph g in order to determine whether g is a clique, a theorem prover must consider a much broader domain for the quantified variables x and y : all terms in \mathcal{L} in the worst case or the more restricted set of all terms of sort node if \mathcal{L} is a sorted logic. In addition, the verification of universally quantified formulas using theorem proving requires the introduction of axioms that articulate the set of objects in the domain of discourse. For a domain containing the objects $\{t_1, t_2, \dots, t_k\}$, an axiom of the form

$$\forall x. \ x = t_1 \vee x = t_2 \vee \dots \vee x = t_k$$

is required. Disjunctive axioms of this type greatly increase the cost of deduction. Attempts to validate universal queries using such axioms resulted in the KADS theorem prover becoming hopelessly lost.

Another advantage of universal attachment relates to *completeness*. As mentioned above, completeness results can be shown for universal attachment provided certain easily-attained closure conditions hold for the system of attachments in use. In contrast, procedural and semantic attachment are inherently incomplete (see [Myers, 1991] for details).

4.2 THEORY RESOLUTION

Theory resolution [Stickel, 1985] is a complete inference rule for integrating theories into a resolution theorem prover without having to resolve directly on the axioms of the theories. Theories can be realized as collections of decision procedures rather than being expressed axiomatically, thus supporting the type of integration required by hybrid systems. In this section, we provide a brief overview of theory resolution along with a comparison between theory resolution and uni-

versal attachment. As we shall see, universal attachment has advantages over theory resolution in terms of generality, modularity and effectiveness of application but lacks theory resolution's ability to handle decision procedures in an elegant fashion.

Standard resolution allows a pair of clauses to be resolved together if a literal occurs in one clause and the negation of that literal occurs in the second clause; in other words, a pair of literals can be selected from the clauses such that the conjunction of those literals is unsatisfiable. With theory resolution, the condition of unsatisfiability is generalized by relativizing unsatisfiability to a theory T ; this condition is referred to as *T -unsatisfiability*. More specifically, a formula ϕ is said to be T -unsatisfiable if $\{\phi\} \cup T$ is unsatisfiable. Integration is achieved by employing programs to determine T -unsatisfiability in much the same way that attached programs are employed by the attachment methods. Theory resolution comes in two varieties. With the simplest form of theory resolution, referred to as *total theory resolution*, a group of clauses can be resolved together if some collection of subclauses from those clauses (called *keys*) can be found whose conjunction is T -unsatisfiable. With *partial theory resolution*, the conjunction of keys need be T -unsatisfiable only when certain *residual conditions* hold.

Universal attachment possesses several features that theory resolution lacks. The first is broad applicability: while theory resolution applies only when the logical component employs a resolution theorem prover, universal attachment can be applied for any form of deduction. Second, theory resolution lacks the translation facilities provided by the attachment and detachment maps. As a result, theory resolution provides less modular connections than universal attachment. Finally, attachment methods accommodate the integration of programs for evaluating logical terms, not just literals or formulas; theory resolution lacks this capability.

The primary advantage of theory resolution over universal attachment is the former's ability to integrate arbitrary decision procedures in a simple way. Due to the substitutional nature of attachments, attachment methods are limited to evaluating subexpressions of a given logical expression. Thus the integration of decision procedures that operate across formulas requires the use of attachments for a metalevel language [Weyhrauch, 1980], an approach whose complexity renders it unappealing. It should be noted, however, that the power that permits theory resolution to handle arbitrary decision procedures in an elegant fashion introduces problems. Because application of the inference rule of theory resolution requires an unsatisfiability test, the inference rule is generally not effective. Even in cases where unsatisfiability can be determined, the use of theory resolution greatly increases the search space of the theorem prover due to

<i>Method</i>	<i>Taxonomy Height</i>					
	2	3	4	5	6	7
UA	.11	.11	.11	.12	.14	.20
KRYPTON	.13	.22	.59	2.85	22.8	235.0
Resolution	.14	.31	.78	2.65	10.3	62.5

Table 1: Subsumption Query Times (in seconds)

the need to recognize appropriate keys (and residual conditions for partial theory resolution). Implementations of theory resolution have focused on special cases in order to mitigate the increased amount of search; we briefly describe the two main approaches here.

Linked Approach

One approach to implementing theory resolution employs the *linked inference* principle [Wos *et al.*, 1984]. A linked inference realization of theory resolution was used to connect the assertional and terminological components in the KRYPTON system [Brachman *et al.*, 1985]. The theorem prover underlying the assertional component of KRYPTON employed a connection graph. Axioms from the terminological theory were compiled into links in the connection graph that grouped collections of T -unsatisfiable literals. A similar realization of theory resolution exists in the Markgraf Karl Refutation Procedure (MKRP) in [Ohlbach and Schmidt-Schauss, 1985].

The biggest drawback of the linked approach is that it relinquishes the ability to exploit black box inference mechanisms. Not surprisingly, the linked approach fares poorly in comparison to universal attachment on problems where efficient nondeductive evaluation mechanisms can be exploited. Table 1 illustrates such a situation, displaying the times required for the two methods to evaluate queries about subsumption relationships in taxonomic structures. These times were obtained using our KADS implementation of universal attachment and a linked implementation of theory resolution contained within KADS (run on a Symbolics 3620); the principal rule of inference used was non-clausal connection-graph resolution. For taxonomies with a fixed branching factor of three but varying heights, proofs were sought that the root of the taxonomy subsumed various leaf nodes. The first row contains the average results for the case where the attachment to *subsumes* (3) was employed. The second entry lists the average results for the linked approach. Although the attached program embodied the same algorithm as described by the logical axioms, the attached approach produced substantially better results. Interestingly enough, unembellished resolution theorem proving fared better than theory resolution when the hierarchy reached a height of six; this phenomenon resulted from the growing overhead required for maintaining appropriate links in the connection graph.

The linked implementation of theory resolution performs well for certain classes of queries. Experiments reported in [Stickel, 1985] show that the method was very successful in solving “Schubert’s Steamroller” puzzle, a noted challenge problem within the theorem proving community. Solving the Steamroller problem involves reasoning about taxonomic relationships in the abstract as opposed to the taxonomic classification of individuals required for the experiment summarized in Table 1. Attachments are of little value in solving problems of this type because the solution requires reasoning that is not procedural in nature; as such, the access to special-purpose evaluation mechanisms provided by attachments is superfluous.

Generalized Resolution

Generalized resolution [Miller and Schubert, 1988; Schubert *et al.*, 1987] is a more flexible realization of theory resolution that readily accommodates black box inference mechanisms. Because generalized resolution realizes a restricted case of theory resolution (2-ary partial theory resolution), the control problems associated with theory resolution are somewhat mitigated. Heuristics for identifying keys are employed in order to constrain applicability of the rule, (*e.g.* “constrain key literals to share arguments”) [Miller, 1988]. However, the search space still remains quite large. Experimentation with generalized resolution has shown that generalized resolution often leads to an overall degradation in system performance [Miller, 1990].

Universal attachment is immune to the control problems that afflict theory resolution because attachment patterns and filter sets clearly delineate the class of logical expressions to which a given attachment applies.

5 TOOLS FOR HYBRID SYSTEMS

Heterogeneous representation systems are inherently more complex than their homogeneous counterparts. For this reason, effective use of hybrid systems will require tools that assist the user in interacting with and controlling them.

Universal attachment affords an advantage over both semantic attachment and theory resolution in this regard, an advantage that derives from the use of universal attachments as ‘building blocks’. Universal attachments draw rich and explicit connections between logical and other types of representations. As such, it is possible to construct tools that manipulate universal attachments in order to support interactions with hybrid frameworks.

One such tool is a method for automatically generating new universal attachments and attached programs, as described in [Myers, 1990]. The method uses compila-

tion techniques to generate new universal attachments and attached programs from a given set of attachments and programs. As an illustration, the method can produce a universal attachment and attached program for the formula

$$\forall xyz. \text{subsumes}(x, z) \wedge \text{subsumes}(y, z) \supset \text{subsumes}(x, y) \vee \text{subsumes}(y, x) \vee \text{eq}(x, y)$$

when given the universal attachment for *subsumes* from Example 2.7 and an appropriate attachment for the equality predicate *eq*. This formula states that a given class has at most one immediate ancestor.

The generation method has proven to be quite effective in practice, providing the means to obtain large collections of attachments and attached programs from small base sets of attachments and programs. Other possible tools for simplifying interactions with hybrid systems include mechanisms for translating between different representation languages and for guaranteeing that different representations of the same information are mutually consistent.

6 CONCLUSIONS

This paper has presented universal attachment, a method for integrating diverse representation and reasoning methods into a logic hybrid. A formal definition of the method was described along with a sound and complete rule of universal attachment inference for use with clausal resolution systems. Comparisons with previous attachment methods showed that universal attachment provides both a broader class of connections and greater modularity. Universal attachment was also shown to have several advantages over theory resolution, including increased modularity and applicability to arbitrary types of theorem proving. Although universal attachment lacks the ability to handle arbitrary decision procedures in the elegant manner of theory resolution, the pattern-directed invocation of universal attachment avoids the control problems inherent to theory resolution.

We hope to extend the ideas presented in this paper towards the development of *more tightly coupled* hybrid systems. By tighter coupling, we mean that the connections between representations are used for tasks that go beyond simplifying expressions in a logical language. The tools described in Section 5 illustrate one form of tighter coupling that can be achieved by reasoning about connections. A second form involves applying the information supplied by connections to the task of controlling inference. For example, it has been shown that alternative representation and reasoning methods can be profitably applied at the metalevel to guide a theorem prover in its search for proofs [Gelernter, 1963]. Although preliminary investigations of this topic have been made by several researchers, it

remains to be seen how the information provided by attachments can be most effectively applied to controlling inference.

Another important area of research involves learning how to most effectively use attachments (and other forms of integration) in order to increase the overall efficiency of a hybrid system. Although a series of preliminary guidelines is presented in [Myers, 1991], the articulation of comprehensive policies for attachment use will require much experimentation.

Acknowledgements

The author is grateful to Adam Grove, Kurt Konolige, Nils Nilsson, Eunok Paek and Carolyn Talcott for discussions about the research reported in this paper. The author would like to thank Mark Stickel for use of the KADS system. The work was supported by an IBM Graduate Fellowship.

References

- [Ait-Kaci and Nasr, 1986] H. Ait-Kaci and R. Nasr. A logic programming language with built-in inheritance. *Journal of Logic Programming*, 3(3):185–215, 1986.
- [Barwise and Etchemendy, 1990] J. Barwise and J. Etchemendy. Visual information and valid reasoning. In W. Zimmerman, editor, *Visualization in Mathematics*. Mathematical Association of America, Washington, DC, 1990.
- [Boyer and Moore, 1981] R. S. Boyer and J. Struther Moore. Metafunctions: Proving them correct and using them efficiently as new proof procedures. In R. S. Boyer and J. Struther Moore, editors, *The Correctness Problem in Computer Science*. Academic Press, 1981.
- [Brachman *et al.*, 1983] R. J. Brachman, R. Fikes, and H. Levesque. KRYPTON: A functional approach to knowledge representation. *IEEE Computer*, 16(10):67–74, 1983.
- [Brachman *et al.*, 1985] R. J. Brachman, V. Gilbert, and H. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts of KRYPTON. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 532–539, 1985.
- [Chang and Lee, 1973] C. Chang and R. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973.
- [Gelernter, 1963] H. Gelernter. Realization of a geometry-theorem proving machine. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*. McGraw-Hill, New York, 1963.
- [Green, 1969] C.C. Green. Application of theorem proving to problem solving. In *Proceedings of the*

- First International Joint Conference on Artificial Intelligence*, pages 219–239, 1969.
- [Malachi et al., 1986] Y. Malachi, Z. Manna, and R. Waldinger. TABLOG: A new approach to logic programming. In D. DeGroot and G. Lindstrom, editors, *Logic Programming: Functions, Relations and Equations*. Prentice-Hall, 1986.
- [Miller and Schubert, 1988] S. Miller and L. Schubert. Using specialists to accelerate general reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 1988.
- [Miller, 1988] S. Miller. Time revisited. Master's thesis, University of Alberta, 1988.
- [Miller, 1990] S. Miller, 1990. Personal communication.
- [Myers, 1990] K. L. Myers. Automatically generating universal attachments through compilation. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990.
- [Myers, 1991] K. L. Myers. *Universal Attachment: A Technique for Integrating Diverse Representation and Reasoning Methods*. PhD thesis, Stanford University, Stanford, Ca, 1991.
- [Ohlbach and Schmidt-Schauss, 1985] H. J. Ohlbach and M. Schmidt-Schauss. The lion and the unicorn. *Journal of Automated Reasoning*, 1(3), 1985.
- [Rich, 1985] C. Rich. The layered architecture of a system for reasoning about programs. In *International Joint Conference on Artificial Intelligence*, pages 540–546, 1985.
- [Robinson and Sibert, 1982] J. Robinson and E. Sibert. LOGLISP: motivation, design and implementation. In K. Clark and S. Tarnlund, editors, *Logic Programming*, pages 299–314. Academic Press, 1982.
- [Russell, 1985] S. Russell. The compleat guide to MRS. Technical Report KSL-85-12, Stanford University, 1985.
- [Schubert et al., 1987] L. Schubert, M. Papalaskaris, and J. Taucher. Accelerating deductive inference: Special methods for taxonomies, colours and times. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*. Springer-Verlag, 1987.
- [Stickel, 1985] M. E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1(4), 1985.
- [Stickel, 1988] M. E. Stickel. The KLAUS automated deduction system. In *Proceedings of the Ninth International Conference on Automated Deduction*, pages 750–751, 1988.
- [Vilain, 1985] M. Vilain. The restricted language architecture of a hybrid representation system. In *International Joint Conference on Artificial Intelligence*, pages 547–551, 1985.
- [Weyhrauch, 1977] R. W. Weyhrauch. A user manual for FOL. Technical Report STAN-CS-77-432, Stanford University, 1977.
- [Weyhrauch, 1980] R. W. Weyhrauch. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, 13:133–170, 1980.
- [Wos et al., 1984] L. Wos, R. Veroff, B. Smith, and W. McCune. The linked inference principle II: the user's viewpoint. In *Seventh International Conference on Automated Deduction*, pages 316–332, Napa, California, 1984.

Belief Revision and Default Reasoning: Syntax-Based Approaches

Bernhard Nebel

German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzengasse 3
D-6600 Saarbrücken 11, Germany
e-mail: nebel@dfki.uni-sb.de

Abstract

Belief revision leads to *temporal nonmonotonicity*, i.e., the set of beliefs does not grow monotonically with time. Default reasoning leads to *logical nonmonotonicity*, i.e., the set of consequences does not grow monotonically with the set of premises. The connection between these forms of nonmonotonicity will be studied in this paper focusing on syntax-based approaches. It is shown that a general form of syntax-based belief revision corresponds to a special kind of partial meet revision in the sense of the theory of epistemic change, which in turn is expressively equivalent to some variants of logics for default reasoning. Additionally, the computational complexity of the membership problem in revised belief sets and of the equivalent problem of derivability in default logics is analyzed, which turns out to be located at the lower end of the polynomial hierarchy.

1 INTRODUCTION

Belief revision is the process of incorporating new information into a knowledge base while preserving consistency. Recently, belief revision has received a lot of attention in AI,¹ which led to a number of different proposals for different applications [Ginsberg, 1986; Ginsberg and Smith, 1987; Dalal, 1988; Gärdenfors and Makinson, 1988; Winslett, 1988; Myers and Smith, 1988; Rao and Foo, 1989; Nebel, 1989; Winslett, 1989; Katsuno and Mendelzon, 1989; Katsuno and Mendelzon, 1990; Doyle, 1990]. Most of this research has been considerably influenced by approaches in philosophical logic, in particular by Gärdenfors and his colleagues [Alchourrón et al., 1985; Gärdenfors, 1988], who developed the *logic of theory*

change, also called *theory of epistemic change*, which will be briefly sketched in Section 2.

Syntax-based approaches to belief revision to be introduced in Section 3 have been very popular because of their conceptual simplicity. However, there also has been criticisms since the outcome of a revision operation relies on arbitrary syntactic distinctions (see, e.g., [Dalal, 1988; Winslett, 1988; Katsuno and Mendelzon, 1989])—and for this reason such operations cannot be analyzed on the *knowledge level*. In [Nebel, 1989] we showed that syntax-based approaches can be interpreted as assigning higher relevance to explicitly represented sentences. Based on that view, one particular kind of syntax-based revision, called *base revision*, was shown to fit into the theory of epistemic change. In Section 4 we generalize this result to *prioritized bases* by employing the notion of *epistemic relevance* [Nebel, 1990]—a complete preorder over the set of all derivable sentences.

The resulting revision operations do not satisfy all *rationality postulates* belief revision operations should obey, however. In Section 5 some interesting special cases of epistemic relevance are analyzed that satisfy all rationality postulates. In particular, we show that *epistemic entrenchment* as introduced in [Gärdenfors and Makinson, 1988] is a special case of *epistemic relevance*.

Makinson and Gärdenfors [1990] showed that there is a tight connection between belief revision and nonmonotonic logics. In Section 6 we will strengthen this result. First, we show that the form of *logical nonmonotonicity* observable when revising beliefs is a *necessary consequence* of *temporal nonmonotonicity* induced by belief revision. Second, we will prove that this similarity can be strengthened to equivalence of expressiveness for particular nonmonotonic logics and belief revision operations in the case of propositional logic. Poole's [1988] and Brewka's [1989; 1990] approaches are shown to be expressively equivalent to some forms of syntax-based belief revision approaches. An interesting consequence of this result is that the “absurd belief state”

¹See also [Brachman, 1990], in which “practical and well-founded theories of belief revision” are called for.

that is inconsistent turns out to be more important than assumed to be in the theory of epistemic change.

Additionally to the logical properties of belief revision and default reasoning, in Section 7 the computational properties are analyzed. As it turns out, propositional syntax-based belief revision and default reasoning is not very much harder than propositional derivability.

2 THE THEORY OF EPISTEMIC CHANGE

In this section we will briefly survey some of the results of the theory of epistemic change in a setting of propositional logic.² Throughout this paper, a propositional language \mathcal{L} with the usual logical connectives ($\neg, \vee, \wedge, \rightarrow$ and \leftrightarrow) is assumed. The countable alphabet of propositional variables a, b, c, \dots is denoted by Σ , propositional sentences by v, w, x, y, z, \dots , constant truth by T , its negation by \perp , and countable sets of propositional sentences by A, B, C, \dots and X, Y, Z, \dots

The symbol \vdash denotes derivability and Cn the corresponding closure operation, i.e.,

$$Cn(A) \stackrel{\text{def}}{=} \{x \in \mathcal{L} \mid A \vdash x\}. \quad (1)$$

Instead of $Cn(\{x\})$, we will also write $Cn(x)$. Deductively closed sets of propositional sentences, i.e., $A = Cn(A)$, are denoted by capital letters from the beginning of the alphabet and are called *belief sets*. Arbitrary sets of sentences are called *belief bases* and are denoted by capital letters from the end of the alphabet. Systems of belief bases and belief sets are denoted by S . Finite belief bases Z are often identified with the conjunction of all propositions $\bigwedge Z$. If $S = \{X_1, \dots, X_n\}$ is a finite family of finite belief bases, then $\bigvee S$ shall denote a proposition logically equivalent to $(\bigwedge X_1) \vee \dots \vee (\bigwedge X_n)$. As usual, we set $\bigvee \emptyset = \perp$.

In [Gärdenfors, 1988] three operations on belief sets are analyzed, namely, *expansion*, *contraction*, and *revision*. *Expansion* is the addition of a sentence x to a belief set A , written $A+x$, resulting in a new (possibly inconsistent) belief set, defined by

$$A+x \stackrel{\text{def}}{=} Cn(A \cup \{x\}). \quad (2)$$

Contraction is the removal of a sentence x from a belief set A resulting in a new belief set, denoted by $A-x$, that does not contain x (if x is not a tautology), and *revision* is the addition of a sentence x to A , denoted by $A+x$, such that $Cn(\perp) \neq A+x$ whenever $\not\vdash \neg x$. Although *contraction* and *revision* are not uniquely determined operations—the only commonly agreed criterion is that the changes to the original belief sets have to be *minimal*—it is possible to constrain

²The formulation in [Alchourrón et al., 1985; Gärdenfors, 1988] is more general in that only some specific properties are required for the underlying logic.

the space of reasonable change operations. Gärdenfors proposed sets of *rationality postulates*³ change operations on belief sets should satisfy. The *Gärdenfors postulates for revision* look as follows (A a belief set, x, y propositional sentences):

- (+1) $A+x$ is a belief set;
- (+2) $x \in A+x$;
- (+3) $A+x \subseteq A+x$;
- (+4) If $\neg x \notin A$, then $A+x \subseteq A+x$;
- (+5) $A+x = Cn(\perp)$ only if $\vdash \neg x$;
- (+6) If $\vdash x \leftrightarrow y$ then $A+x = A+y$;
- (+7) $A+(x \wedge y) \subseteq (A+x)+y$;
- (+8) If $\neg y \notin A+x$,
then $(A+x)+y \subseteq A+(x \wedge y)$.

These postulates intend to capture the intuitive meaning of minimal change—from a logical point of view [Alchourrón et al., 1985; Gärdenfors, 1988]. (+1) states that revision of belief set always results in a belief set. (+2) formalizes the requirement that revision is always successful. (+3) gives an upper bound for a revised belief set. It should at most contain the consequences of the original belief set and the new sentence. (+4) is the conditional converse of (+3). In case when x is consistent with the original belief set, the revised belief set shall at least contain the original belief base and the new sentence. (+5) states that inconsistency should be avoided when possible, and (+6) formalizes the requirement that revision shall be independent from the syntactic form of the sentence the belief set is revised by. While the first six postulates, also called *basic postulates*, are straightforward, the last two postulates are less obvious. They can be interpreted as generalizations of (+3) and (+4).

Based on this framework, it is possible to analyze different ways of defining revision operations. In [Alchourrón et al., 1985], so-called partial meet revisions are investigated. This notion is based on systems of maximal (w.r.t. to set-inclusion) subsets of a given belief set A that do not allow the derivation of x , called the *removal of x* and written $A \downarrow x$:

$$A \downarrow x \stackrel{\text{def}}{=} \{B \subseteq A \mid B \not\vdash x, \forall C: B \subset C \subseteq A \Rightarrow C \vdash x\}. \quad (3)$$

A *partial meet revision* (on A for all x) is defined by a *selection function* S that selects a nonempty subset of $A \downarrow \neg x$ (provided $A \downarrow \neg x$ is nonempty, \emptyset otherwise) in

³In order to avoid confusion, one should note that *rationality* in the sense of the theory of epistemic change means an *idealization*: “In this way the rationality criteria serve as regulative ideals. Actual psychological states of belief normally fail to be ideally rational in this sense” [Gärdenfors, 1988, Section 1.2]. Further, this notion of *rationality* is quite different from the notion of economic rationality [Doyle, 1990].

the following way:⁴

$$A + x \stackrel{\text{def}}{=} (\bigcap S(A \downarrow \neg x)) + x. \quad (4)$$

Such partial meet revisions satisfy unconditionally the first six postulates, also called *basic postulates*. Furthermore, it is possible to show that all revision operations satisfying the basic postulates are partial meet revisions [Gärdenfors, 1988, Theorem 4.13]. Actually, this and the other results cited below were proven for contraction. However, if contraction and revision satisfy the basic postulates, they are interdefinable by the Harper (5) and Levi (6) identity:

$$A - x = (A + \neg x) \cap A, \quad (5)$$

$$A + x = (A - \neg x) + x. \quad (6)$$

Further, the eight *Gärdenfors postulates for contraction* (see, e.g. [Alchourrón *et al.*, 1985; Gärdenfors, 1988]) are equivalent to the revision postulates under these definitions in the following sense. The first six, seven, or eight contraction postulates are satisfied if and only if the first six, seven or eight, revision postulates for the corresponding revision operation are satisfied, respectively [Gärdenfors, 1988, Theorem 3.2–3.5].

It should be noted that two special cases of partial meet revisions are unreasonable [Alchourrón *et al.*, 1985; Gärdenfors, 1988]. The first special case is that S always selects all of the elements of $A \downarrow \neg x$ —leading to the so-called *full meet revision*. In this case $A + x = Cn(x)$ if $\neg x \in A$. Although unreasonable, full meet revision is “fully rational” in the sense that it satisfies all the Gärdenfors postulates, as is easy to verify.

The second special case is that S always selects *singles*tions from $(A \downarrow \neg x)$ —resulting in the class of so-called *maxi-choice revisions*. These revision operations have the property that $A + x$ is a *complete* belief set—provided that $\neg x \in A$. This means $y \in A + x$ or $\neg y \in A + x$ for every $y \in \mathcal{L}$. In other words maxi-choice revisions lead to an unmotivated inflation of beliefs.

3 SYNTAX-BASED REVISION APPROACHES

The theory sketched above captures the *logical* portion of minimal change giving us a kind of yardstick to evaluate approaches to belief revision. However, it still leaves open the problem of how to specify additional restrictions so that a revision operation also satisfies a “pragmatic” measure of minimal change.

Two principal points of departure are conceivable. Starting with a belief base as the representation of

⁴Note that all elements of $A \downarrow \neg x$ are belief sets and that the intersection of belief sets is a belief set again.

a belief set, either the syntactic form of the belief base [Fagin *et al.*, 1983; Ginsberg, 1986; Nebel, 1989] or the possible states of the world described by the belief base—the models of the belief base—could be changed minimally [Dalal, 1988; Winslett, 1988; Katsuno and Mendelzon, 1989; Katsuno and Mendelzon, 1990]. The former approach seems to be more reasonable if the belief base corresponds to a body of explicit beliefs that has some relevance, such as a code of norms or a scientific or naive theory which is almost correct. The latter view seems plausible if the application is oriented towards minimal change of the state of the world described by a belief set.

The idea of *changing a description minimally* could be formalized by selecting maximal subsets of the belief base not implying a given sentence. If there is more than one such maximal subset, the intersection of the consequences of these subsets is used as the result. Thus, using $(Z \downarrow x)$ as the set of maximal subsets of Z not implying x as above, *simple base revision*, written as $Z \oplus x$, could be defined as follows [Fagin *et al.*, 1983; Ginsberg, 1986; Nebel, 1989]:

$$Z \oplus x \stackrel{\text{def}}{=} \left(\bigcap_{Y \in (Z \downarrow x)} Cn(Y) \right) + x. \quad (7)$$

The operation \oplus considers all sentences in a base as equally relevant. In most applications, however, we want to distinguish between the relevance of sentences (see, e.g., [Fagin *et al.*, 1983; Ginsberg, 1986]). For this purpose, we assume that Z is partitioned into disjoint *prioritized classes* Z_i , $i \geq 1$, and define the *prioritized removal* of x , written $Z \Downarrow x$, in a way such that sentences in Z_i have higher priority than those in Z_j iff $i < j$:

$$Z \Downarrow x \stackrel{\text{def}}{=} \left\{ Y \subseteq Z \mid Y \not\vdash x, Y = \bigcup_{i \geq 1} Y_i, \begin{array}{l} \forall i \geq 1: (Y_i \subseteq Z_i, \\ \forall X: Y_i \subset X \subseteq Z_i \Rightarrow (\bigcup_{j=1}^{i-1} Y_j \cup X) \vdash x) \end{array} \right\}. \quad (8)$$

Intuitively, the elements of $Z \Downarrow x$ are constructed in a stepwise manner starting with Z_1 and selecting as many sentences from Z_i as possible such that the selected sentences from Z_1, \dots, Z_i do not lead to the derivation of x .

Using \Downarrow instead of \downarrow in the definition (7) leads then to a *prioritized base revision* operation, denoted by $\hat{\oplus}$, with the special case of only one priority class that is identical with simple base revision.

In the interesting special case when we are dealing with *finite* belief bases, the result of a base revision can be finitely represented.

Proposition 1 *If Z is a finite belief base then*

$$Z \hat{\oplus} x = Cn((\bigvee (Z \Downarrow \neg x)) \wedge x), \quad (9)$$

for every prioritized base revision $\hat{\oplus}$ on Z .⁵

In order to demonstrate how base revision works, let us assume the following scenario. Assume that a suspect tells you that he went to the beach for swimming and assume that you have observed that the sun was shining. Further, you firmly believe that going to the beach for swimming when the sun is shining implies a sun tan. If you then discover that the suspect is not tanned, there is an inconsistency to resolve. Supposing the following propositions:

$$\begin{aligned} b &= \text{"going to the beach for swimming"}, \\ s &= \text{"the sun is shining"}, \\ t &= \text{"sun tan"}, \end{aligned}$$

the situation can be modeled formally by a prioritized base Z :

$$\begin{aligned} Z_1 &= \{(b \wedge s \rightarrow t)\}, \\ Z_2 &= \{s\}, \\ Z_3 &= \{b\}, \\ Z &= Z_1 \cup Z_2 \cup Z_3. \end{aligned}$$

From this belief base t can be derived. If we later observe that $\neg t$, the belief base has to be revised:

$$\begin{aligned} Z \hat{\oplus} \neg t &= \bigcap (Cn(Z \downarrow t)) + \neg t \\ &= Cn(\bigvee \{(b \wedge s \rightarrow t), s\}) + \neg t \\ &= Cn(\{(b \wedge s \rightarrow t), s, \neg t\}). \end{aligned}$$

In particular, we would conclude that b was a lie.

A consequence of the definition of (simple and prioritized) base revision is that for two different belief bases X and Y that have the same meaning, i.e., $Cn(X) = Cn(Y)$, base revision can lead to different results, i.e., $Cn(X \hat{\oplus} x) \neq Cn(Y \hat{\oplus} x)$. Base revision has a "morbid sensitivity to the syntax of the description of the world" [Winslett, 1988], which is considered as an undesirable property. It is argued that revision shall be independent from the syntactical representation of a belief set, that they should be specified on the *knowledge level* [Newell, 1982]. Dalal [1988] formulated the *principle of irrelevance of syntax* which states that a revision shall be independent of the syntactic form of the belief base representing a belief set and of the syntactic form of the sentence that has to be incorporated into the belief set (see also [Katsuno and Mendelzon, 1989]).

Obviously, base revision does not satisfy the principle of irrelevance of syntax—and is not a revision operation in the sense of the theory of epistemic change for this reason. Worse yet, abstracting from the syntactic representation of a belief base and considering the logical equivalent belief set leads to nowhere.

⁵Full proofs for this and the following propositions can be found in [Nebel, 1991].

Simple base revision applied to belief sets is equivalent to full meet revision, thus, useless. For these reasons, it is argued in [Dalal, 1988; Winslett, 1988; Katsuno and Mendelzon, 1989] that revision shall be performed on the *model-theoretic* level, i.e., by viewing a belief set as the set of models that satisfy a given belief base and by performing revision in a way that selects models that satisfy the new sentence and *differ minimally* from the models of the original belief base. In order to define what the term *minimal difference* means, we have to say something about how models are to be compared, though. In Dalal [1988], for instance, the "distance" between models is measured by the number of propositional variables that have different truth values. Katsuno and Mendelzon [1989] generalize this approach by considering particular orderings over models. In any case, it is impossible to define a revision operation by referring only to logical properties. Some inherently extra-logical, pragmatic preferences are necessary to guide the revision process.

As argued above, for some applications it does not seem to be a bad idea to derive preferences from the syntactic form of the representation of a belief set. Actually, from a more abstract point of view, it is not the particular syntactic form a belief base we are interested in, but it is the fact that we believe that a particular set of sentences is more valuable or justified than another logically equivalent set, and we want to preserve as many of the "valuable" sentences as possible. Using this idea it is possible to reconstruct base revision in the framework of the theory of epistemic change by employing the notion of *epistemic relevance* [Nebel, 1990].

4 EPISTEMIC RELEVANCE AND BASE REVISION

The intention behind base revision is that all the sentences in a belief base X are considered as *relevant*—some perhaps more so than others. For this reason we want to give up as few sentences from X as possible, while with sentences that are only derivable we are more liberal. This idea can be formalized by employing a *complete preorder* with maximal elements, written $x \preceq y$, on the elements of a belief set. In other words, we consider a *reflexive* and *transitive* relation on $Cn(X)$ such that for all $x, y \in Cn(X)$ we have $x \preceq y$ or $y \preceq x$. Further, there exists at least one maximal element x , i.e., for no element y : $x \preceq y$ and $y \not\preceq x$. This relation will be called *epistemic relevance ordering*. It induces an equivalence relation, written $x \simeq y$, as follows:

$$x \simeq y \quad \text{iff} \quad (x \preceq y \text{ and } y \preceq x). \quad (10)$$

The corresponding equivalence classes are denoted by \bar{x} and are called *degrees of epistemic relevance*. Since the preorder is complete, \preceq is a linear order on the

degrees of epistemic relevance. Further, there exists a maximal such degree because the preorder contains maximal elements. Using the degrees of epistemic relevance, we define a *strict partial ordering* expressing preferences on subsets $X, Y \in 2^A$, written as $X \ll Y$, by

$$X \ll Y \text{ iff } \exists \bar{v}: \left((X \cap \bar{v} \subset Y \cap \bar{v}) \text{ and } \forall \bar{w} \not\leq \bar{v}: (X \cap \bar{w} = Y \cap \bar{w}) \right), \quad (11)$$

which in turn can be used to define a selection function S_{\leq} that selects all maximally preferred elements of $A \downarrow x$:

$$S_{\leq}(A \downarrow x) \stackrel{\text{def}}{=} \{B \in (A \downarrow x) \mid \forall C \in (A \downarrow x): B \not\ll C\}. \quad (12)$$

Such a selection function may then be used to define a revision operation as done in equation (4). Revisions defined in this way will be called *revisions based on epistemic relevance*. Analyzing the properties of such revisions, we note that they satisfy most of the Gärdenfors postulates.

Theorem 2 *Revisions based on epistemic relevance satisfy (1)–(7).*

Proof Sketch: First of all, note that any revision based on epistemic relevance is a partial meet revision because S_{\leq} selects always a nonempty subset of $(A \downarrow x)$, provided this set is nonempty. By Theorem 4.13 of [Gärdenfors, 1988], any partial meet revision satisfies (1)–(6). Satisfaction of (7) follows from the fact that S_{\leq} is a *relational* selection function [Gärdenfors, 1988, Lemma 4.14]. ■

Note that revision based on epistemic relevance does not satisfy (8) in general. The interesting point about such partial meet revisions is that they correspond to prioritized base revision as defined in Section 3. The deeper reason for this correspondence is that a selection function can be constructed in a way such that it selects the intersection of the consequential closure of a system of sets, as spelled out below.

Lemma 3 *Let A be a belief set, and let x be a sentence such that $\neg x \in A$. Let S be a system of subsets of A , where $Z \not\models \neg x$, for all $Z \in S$. Then*

$$\begin{aligned} \left(\bigcap \{C \in A \downarrow \neg x \mid \exists Z \in S: Z \subseteq C\} \right) + x &= \\ &= \left(\bigcap_{Z \in S} C_n(Z) \right) + x. \end{aligned} \quad (13)$$

Using this lemma, the correspondence between revision based on epistemic relevance and prioritized base revision can be easily shown.

Theorem 4 *For any revision operation $\dot{+}$ defined on A and based on epistemic relevance there exists a corresponding prioritized base revision $\hat{\oplus}$ on some base Z and vice versa such that*

$$A \dot{+} x = Z \hat{\oplus} x. \quad (14)$$

Proof Sketch: For the limiting cases $\neg x \notin A$ or $\vdash \neg x$ the theorem holds trivially.

\Rightarrow : First, any belief set is a belief base by definition. Second, any epistemic entrenchment ordering defines priority classes on such a base. Third, for any deductively closed prioritized base A , a set X is a member of $(A \Downarrow x)$ if and only if X is maximal in $(A \downarrow x)$ w.r.t. \ll . Applying Lemma 3, the conclusion follows.

\Leftarrow : Given a prioritized base Z , define an epistemic entrenchment ordering on $C_n(Z)$ such that $v \leq w$ iff $v \in C_n(Z) - Z$ or $v \in Z_i$, $w \in Z_j$, and $i \geq j$. Now it is easy to show that $B \in S_{\leq}(C_n(Z) \downarrow x)$ iff there is a set $X \in (Z \Downarrow x)$ and $X \subseteq B$. Applying again Lemma 3, shows that a revision based on the relevance ordering defined above leads to the same results as the prioritized base revision. ■

This means that prioritized base revision coincides with revision based on epistemic relevance. This abstract view on syntax-based revision may also answer some of the questions raised by Myers and Smith [1988]. They observed that sometimes base revision does not seem to be the appropriate operation because some derived information turns out to be more relevant than the syntactically represented sentences in a belief base, and we get the wrong results when using base revision. However, there is no magic involved here. Base revision leads to the right results only if the syntactic representation really reflects the epistemic relevance. For this reason, the notion of revision based on epistemic relevance seems to be preferable over base revision because it avoids the confusion between surface-level syntactic representation and the intended relevance of propositions.

The question of whether the above defined correspondence can be exploited computationally cannot be answered positively in the general case. For the case of belief sets that are finite modulo logical equivalence, however, revision based on epistemic relevance can be performed by a prioritized base revision on a finite base.

Proposition 5 *Let A be a belief set finite modulo logical equivalence. If $\dot{+}$ is a revision based on epistemic relevance defined on A , then there exists a finite prioritized base Z , such that for all x :*

$$A \dot{+} x = Z \hat{\oplus} x. \quad (15)$$

Although revisions based on epistemic relevance do not satisfy all Gärdenfors postulates, there are special cases that do so. A trivial special case is a revision based on only one degree of epistemic relevance, which is equivalent to full meet revision. There are more interesting cases, however, we will investigate in the following section.

5 EPISTEMIC RELEVANCE AND ENTRENCHMENT

Gärdenfors and Makinson claim that the notion of *epistemic entrenchment* introduced in [Gärdenfors and Makinson, 1988] is closely related to the notion of *database priorities* as proposed in [Fagin *et al.*, 1983]. Since the notion of database priorities is identical to the notion of *priority classes* introduced in Section 3, which in turn coincides with *degrees of epistemic relevance*, one would expect that epistemic entrenchment orderings are closely related to epistemic relevance orderings. Although the intuitions are clearly similar, the question is whether the different formalizations lead to identical result of revision operations.

Epistemic entrenchment orderings, written as $x \preceq_e y$, are defined over the entire set of sentences \mathcal{L} and have to satisfy the following properties:

- (\preceq_e 1) If $x \preceq_e y$ and $y \preceq_e z$, then $x \preceq_e z$.
- (\preceq_e 2) If $x \vdash y$, then $x \preceq_e y$.
- (\preceq_e 3) For any x, y , $x \preceq_e (x \wedge y)$ or $y \preceq_e (x \wedge y)$.
- (\preceq_e 4) When $A \neq Cn(\perp)$, then $x \notin A$ iff $x \preceq_e y$ for all $y \in \mathcal{L}$.
- (\preceq_e 5) If $y \preceq_e x$ for all $y \in \mathcal{L}$, then $\vdash x$.

Using such a relation, Gärdenfors and Makinson define *contraction based on epistemic entrenchment*, written $A \trianglelefteq x$, by

$$y \in A \trianglelefteq x \text{ iff } y \in A \text{ and } ((x \vee y) \not\preceq_e x \text{ or } \vdash x) \quad (16)$$

and show that a contraction based on epistemic entrenchment satisfies all rationality postulates for contraction as well as the following condition [Gärdenfors and Makinson, 1988, Theorem 4]:

$$x \preceq_e y \text{ iff } x \notin A \trianglelefteq (x \wedge y) \text{ or } \vdash (x \wedge y). \quad (17)$$

Further, they show that any contraction operation satisfying all of the rationality postulates is generated by some epistemic entrenchment ordering [Gärdenfors and Makinson, 1988, Theorem 5].

The question is now how to interpret these results in the framework of epistemic relevance. It follows straightforwardly that the restriction of \preceq_e to the sentences in a belief set can be considered as an epistemic relevance ordering as defined in the previous section. Further, in this case, using interdefinability of revision and contraction, definition (16) coincides with a contraction operation that is defined by using the Harper identity (5) and a revision operation based on a epistemic relevance ordering that satisfies (\preceq_e 1)–(\preceq_e 5).

Theorem 6 Suppose a belief set A , an epistemic entrenchment ordering \preceq_e , and a contraction operation \trianglelefteq based on \preceq_e . Let \preceq be an epistemic relevance ordering that is the restriction of \preceq_e to A , and let $\dot{+}$ be a

revision based on the epistemic relevance ordering \preceq . Then

$$A \trianglelefteq x = (A \dot{+} \neg x) \cap A. \quad (18)$$

Proof Sketch: For the limiting cases $\vdash x$ and $x \notin A$ the theorem holds trivially. For the principal case, one straightforwardly verifies that

$$y \in \bigcap \mathcal{S}_{\preceq}(A \downarrow x) \text{ iff } y \in A \text{ and } (x \vee y) \not\preceq_e x. \quad (19)$$

From this the conclusion follows immediately. ■

Thus, the notion of epistemic entrenchment can indeed be viewed as a refined special case of database priorities or epistemic relevance. It is not obvious, however, how to arrive at such epistemic entrenchment orderings. While epistemic relevance can be easily derived from given priority classes, it is not clear whether there are natural ways to generate epistemic entrenchment orderings. In [Gärdenfors and Makinson, 1988] it is proposed to start with a complete ordering over the maximal disjunctions derivable from a belief set. Despite the fact that this does not sound very “natural”, it also implies that a large amount of information has to be supplied, sometimes too much (see Section 7), in order to change a belief set.

Interestingly, there is another special case of epistemic relevance that leads to a revision operation that satisfies all postulates. When all priority classes of a prioritized belief base are singletons, then the prioritized base revision (as well as the corresponding partial meet revision and the epistemic relevance ordering) are called *unambiguous*.

Proposition 7 Let Z be a prioritized belief base such that all priority classes are singletons. Then $(Z \downarrow x)$ is a singleton iff $\vdash x$.

Clearly, the corresponding epistemic relevance ordering is not necessarily an epistemic entrenchment ordering. Nevertheless, unambiguous partial meet revisions satisfy all rationality postulates.

Theorem 8 Let \preceq be an unambiguous epistemic relevance ordering. Then the revision based on this ordering satisfies all Gärdenfors postulates.

Proof Sketch: By Theorem 2 (4)–(7) are satisfied. If $\neg x \notin A$, then (8) holds because of (4). If $\vdash \neg x$, then (8) holds because the “if” part of (8) is never true. For the principal case we make use of Proposition 7 and Theorem 4 and show that for the unambiguously prioritized base Z :

$$\begin{aligned} & \text{If } \{X\} = (Z \downarrow \neg x) \text{ and} \\ & X \cup \{x\} \not\vdash \neg y \text{ and} \\ & \{Y\} = (Z \downarrow (\neg x \vee \neg y)) \\ & \text{then } X \subseteq Y. \end{aligned} \quad (20)$$

Assume that (20) is violated, i.e., there exists a smallest index j such that $X \cap \bigcup_{i=1}^j Z_i \supset Y \cap \bigcup_{i=1}^j Z_i$. This, however, would mean that $X \supseteq ((Y \cap \bigcup_{i=1}^j Z_i) \cup Z_j) \vdash \neg x \vee \neg y$. Thus, $X \vdash x \rightarrow \neg y$, which is equivalent to $X \cup \{x\} \vdash \neg y$ contradicting the precondition in (20). ■

Although an unambiguous relevance ordering is not necessarily an entrenchment ordering, using (17) it is possible to specify an epistemic entrenchment ordering that leads to an identical revision operation since unambiguous revisions are fully rational.

6 BELIEF REVISION AND DEFAULT REASONING

Doyle remarked in [Doyle, 1990, App. A] that “the adjective ‘nonmonotonic’ has suffered much careless usage recently in artificial intelligence, and the only thing common to many of its uses is the term ‘nonmonotonic’ itself.” Doyle identified two principal ideas behind the use of this term, namely,

[...] that attitudes are gained and lost over time, that reasoning is nonmonotonic—this we call *temporal* nonmonotonicity—and that unsound assumptions can be the deliberate product of sound reasoning, incomplete information, and a “will to believe”—which we call *logical* nonmonotonicity.

Although these two forms of nonmonotonicity should not be confused, they are intimately connected. In particular, the temporal nonmonotonicity induced by belief revision, i.e., the fact that in general we do not have $A \subseteq A + x$, is connected with logical nonmonotonicity induced by some forms of default reasoning.

When reasoning with defaults in a setting as described in [Poole, 1988; Brewka, 1989], we are prepared to “drop” some of the defaults if they are inconsistent with the facts. This, however, is quite similar to what we are doing when revising beliefs in the theory of epistemic change. Propositions of a theory are given up when they are inconsistent with new facts. Since default reasoning leads to logical nonmonotonicity, one would expect that belief revision is nonmonotonic in the facts to be added, i.e., we would expect that $Cn(x) \subseteq Cn(y)$ does not imply $A + x \subseteq A + y$. Conversely, requiring monotony in the second operand of a belief revision operation is impossible in the general case.

Proposition 9 Let $\dot{+}$ be a belief revision operation defined on a belief set A . If for all x, y

$$A + x \subseteq A + y \quad \text{if } Cn(x) \subseteq Cn(y), \quad (21)$$

then

1. $A = Cn(\emptyset)$ and $A + x = Cn(x)$, or
2. $A = Cn(\perp)$ and $A + x = Cn(x)$, or
3. $\dot{+}$ violates at least one of the basic Gärdenfors postulates.

Makinson and Gärdenfors [1990] use this similarity of logical nonmonotonicity and the nonmonotonicity of belief revision in the second operand as a starting point to investigate the relationship between nonmonotonic logics and belief revision on an very general and abstract level. They compare various general conditions on nonmonotonic provability relations with the Gärdenfors postulates.

For the approaches to belief revision described in the previous section there is an even stronger connection to nonmonotonic logic. Prioritized base revision, and hence partial meet revision based on epistemic relevance, is expressively equivalent to *skeptical provability*⁶ in Poole’s [1988] *theory formation* approach and Brewka’s [1989] *level default theories* (LDT)—for the case of finitary propositional logic.

A common generalization of both approaches are *ranked default theories* (RDT). A RDT Δ is a pair $\Delta = (\mathcal{D}, \mathcal{F})$, where \mathcal{D} is a finite sequence $\langle \mathcal{D}_1, \dots, \mathcal{D}_n \rangle$ of finite sets of sentences (propositional, in our case) interpreted as ranked defaults and \mathcal{F} is a finite set of sentences interpreted as hard facts.

An *extension* of Δ is a deductively closed set of propositions $E = Cn((\bigcup_{i=1}^n \mathcal{R}_i) \cup \mathcal{F})$ such that for all i with $1 \leq i \leq n$:

1. $\mathcal{R}_i \subseteq \mathcal{D}_i$,
2. $(\bigcup_{j=1}^i \mathcal{R}_j) \cup \mathcal{F}$ is consistent,⁷ and
3. $(\bigcup_{j=1}^i \mathcal{R}_j)$ is set-inclusion maximal.

A sentence x is *strongly provable* in Δ , written $\Delta \vdash x$, iff for all extensions E of Δ : $x \in E$.

Poole’s approach is a special case of RDT’s where $\mathcal{D} = \langle \mathcal{D}_1 \rangle$, and Brewka’s LDT’s are RDT’s with $\mathcal{F} = \emptyset$. Note, however, that the expressive difference between RDT’s and LDT’s is actually very small and shows up only if \mathcal{F} is inconsistent. In this case, RDT’s allow the derivation of \perp while this is impossible in LDT’s.

Theorem 10 Let $\Delta = (\langle \mathcal{D}_1, \dots, \mathcal{D}_n \rangle, \mathcal{F})$ be a RDT. Let $Z = \bigcup_{i=1}^n \mathcal{D}_i$ be a prioritized base with priority classes $\mathcal{D}_1, \dots, \mathcal{D}_n$. Then

$$\Delta \vdash x \quad \text{iff } x \in (Z \hat{\oplus} \mathcal{F}). \quad (22)$$

⁶A correspondence to *credulous* derivability can be achieved if a notion of *nondeterministic* revision as proposed in [Doyle, 1990] is adopted.

⁷Note that this definition, which is similar to the definition of an extension in [Poole, 1988], excludes inconsistent extensions. Nevertheless, the definition of strong provability implies that \perp can be derived iff \mathcal{F} is inconsistent.

Proof Sketch: In the limiting case when $\mathcal{F} \vdash \perp$, $Z \dot{\oplus} \mathcal{F} = Cn(\perp)$. Further, in this case there is no extension of Δ , hence $\Delta \not\vdash x$ for all $x \in \mathcal{L}$ by the definition of strong provability.

When \mathcal{F} is consistent, it is easy to see that E is an extension of Δ if and only if there is a belief base $X \in (Z \Downarrow \neg(\wedge \mathcal{F}))$ such that $E = Cn(X \cup \mathcal{F})$. ■

This means that finite ranked default theories have the same expressive power as prioritized base revision operations, which coincide with revisions based on epistemic relevance by Theorem 4.

It should be noted that in ranked default theories there is no requirement on the *internal consistency* of defaults. This means that the set $\bigcup_i \mathcal{D}_i$ may very well be inconsistent. In Theorem 10 that may lead to $\perp \in Cn(Z)$, i.e., the belief set to be revised is inconsistent. Although this might sound unreasonable in the context of modeling (idealized) epistemic states—in fact, inconsistency is indeed explicitly excluded by requirement (2.2.1) in [Gärdenfors, 1988]—it does not lead to technical problems in the theory of epistemic change. Additionally, it is possible to give a transformation between reasoning in RDT's and prioritized base revision using only consistent belief sets.

Corollary 11 *Let Δ be a RDT as above. Then there exists a consistent prioritized base Z and a proposition y such that*

$$\Delta \vdash x \text{ iff } x \in (Z \dot{\oplus} (y \wedge \mathcal{F})). \quad (23)$$

Proof Sketch: Define Z as in Theorem 10. Transform every sentence in Z into negation normal form (i.e., into a formula such that negation signs appear only in front of propositional variables), replace any negative literal $\neg a$ in all sentences of Z by a fresh variable a' and define (assuming w.l.g. that Σ is finite):

$$y \stackrel{\text{def}}{=} \bigwedge_{a \in \Sigma} (\neg a \leftrightarrow a'). \quad (24)$$

This ensures that Z is consistent and that the result of the revision is the same as in Theorem 10. ■

From the results above and the translation of (+8) to a condition on nonmonotonic derivability relations in [Makinson and Gärdenfors, 1990], it follows that the derivability relation of RDT's w.r.t. the set of hard facts \mathcal{F} does not satisfy the condition of *rational monotony* (see [Makinson and Gärdenfors, 1990]). Note that this result depends on the *exact* correspondence between RDT's and belief revision based on epistemic relevance. In [Makinson and Gärdenfors, 1990; Gärdenfors, 1990] the correspondence between Poole's logic and belief revision was only approximate because the defaults were assumed to be deductively closed.

Another interesting observation in this context is that the addition of *constraints* to RDT's is similar but not

identical to a contraction operation as defined in Section 2. Poole [1988] introduced *constraints*—another set of sentences—as a means to restrict the applicability of defaults. A *ranked default theory with constraints* is a triple $\Delta = (\mathcal{D}, \mathcal{F}, \mathcal{C})$, where \mathcal{D} and \mathcal{F} are defined as above and \mathcal{C} is a finite set of sentences interpreted as constraints. The notion of an extension is modified as follows. Instead of condition 2. it is required that

$$2. (\bigcup_{i=1}^n \mathcal{R}_i) \cup \mathcal{F} \cup \mathcal{C} \text{ is consistent.}$$

Provided the set $\mathcal{F} \cup \mathcal{C}$ is consistent, which is the interesting case, skeptical derivability can be modeled as a form of contraction on *belief bases* (see [Nebel, 1989]), followed by an expansion.

Theorem 12 *Let $\Delta = ((\mathcal{D}_1, \dots, \mathcal{D}_n), \mathcal{F}, \mathcal{C})$ be an RDT with constraints such that $\mathcal{F} \cup \mathcal{C}$ is consistent. Let $Z = \bigcup_{i=1}^n \mathcal{D}_i$ be a prioritized base with $\mathcal{D}_1, \dots, \mathcal{D}_n$ the priority classes of Z . Then*

$$\Delta \vdash x \text{ iff } \bigvee (Z \Downarrow \neg(\mathcal{F} \wedge \mathcal{C})) \wedge \mathcal{F} \vdash x \quad (25)$$

Proof Sketch: Assuming that $\mathcal{F} \cup \mathcal{C}$ is consistent, any extension E of Δ is the set of consequences of \mathcal{F} and a set $X = \bigcup_{i=1}^n \mathcal{R}_i$ such that for all $j \leq n$: $\bigcup_{i=1}^j \mathcal{R}_i$ is consistent with $\mathcal{F} \cup \mathcal{C}$ and maximal. This, however, is by definition equivalent to $X \in (Z \Downarrow \neg(\mathcal{F} \wedge \mathcal{C}))$. ■

This means that contrary to the opinion that there is no counter-part to contraction in nonmonotonic logics as spelled out in [Makinson and Gärdenfors, 1990], default reasoning with constraints in Poole's theory formation approach can be modeled by using an operation similar to contraction. However, this similarity does not apply to contraction on belief sets. Assuming that the defaults are internally inconsistent, i.e., $Z \vdash \perp$, application of the Harper identity (5) leads to

$$A \dashv \neg x = A + x, \quad (26)$$

which means that the constraints become part of every extension violating the intention behind introducing constraints. This does not happen when contracting a base because base contraction removes more beliefs than contraction on belief sets (see also [Nebel, 1989]).

7 COMPUTATIONAL COMPLEXITY

For the investigation of the computational complexity of belief revision, we consider the problem of determining membership of a sentence y in a belief set $A = Cn(Z)$ revised by x , i.e.,

$$y \in A + x. \quad (27)$$

As the input size we use the sum of the size $|Z|$ of the belief base Z that represents A and the sizes $|x|$ and $|y|$ of the sentences x and y , respectively.

This assumption implies that the representation of the preference relation used to guide the revision process should be polynomially bounded by $|Z| + |x| + |y|$. Although this sounds like a reasonable restriction, it is not met by all belief revision approaches. Belief revision based on *epistemic entrenchment* orderings [Gärdenfors and Makinson, 1988], for instance, requires more preference information in the general case. An epistemic entrenchment ordering over all elements of a belief set can be uniquely characterized by an *initial* complete order over the set of all derivable *maximal disjunctions* (over all literals) [Gärdenfors and Makinson, 1988, Theorem 7]. This set is logarithmic in the size of the set of formulas (modulo logical equivalence) in a *belief set*. However, the number of maximal disjunctions may still be very large, namely, exponential in the size of a *belief base*.

A similar statement could be made about revisions based on epistemic relevance. However, if we consider only complete preorders over Z with the understanding that the degree of least relevant sentences is just $Cn(Z) - Z$, then the ordering is represented in a way that is polynomially bounded by $|Z|$ and $\dot{+}$ can be computed by using the corresponding prioritized base revision.

Analyzing the computational complexity of the belief revision problems, the first thing one notes that deciding the trivial case $y \in Cn(\emptyset) \dot{+} x$ is already co-NP-complete,⁸ and we might give up immediately. However, finding a characterization of the complexity that is more fine grained than just saying it is NP-hard can help to understand the structure of the problem better. In particular, we may be able to compare the inherent complexity of different approaches and, most importantly, we may say something about feasible implementations, which most likely will make compromises along the line that the expressiveness of the logical language is restricted and/or incompleteness is tolerated at some point. For this purpose we have to know, however, what the sources of complexities are.

The belief revision problems considered in this paper fall into complexity classes located at the lower end of the *polynomial hierarchy*. Since this notion is not as common as the central complexity classes, it will be briefly sketched [Garey and Johnson, 1979, Sect. 7.2]. Let X be a class of decision problems. Then P^X denotes the class of decision problems $L \in P^X$ such that there is a decision problem $L' \in X$ and a polynomial Turing-reduction from L to L' , i.e., all instances of L can be solved in polynomial time on a Turing machine that employs an oracle for L' . Similarly, NP^X denotes the class of decision problems $L \in NP^X$ such that there is nondeterministic Turing-machine that solves all in-

⁸We assume some familiarity with the basic notions of the theory of NP-completeness as presented in the first few chapters of [Garey and Johnson, 1979].

stances of L in polynomial time using an oracle for $L' \in X$. Based on these notions, the sets Δ_k^p , Σ_k^p , and Π_k^p are defined as follows:⁹

$$\Delta_0^p = \Sigma_0^p = \Pi_0^p = P, \quad (28)$$

$$\Delta_{k+1}^p = P^{\Sigma_k^p}, \quad (29)$$

$$\Sigma_{k+1}^p = NP^{\Sigma_k^p}, \quad (30)$$

$$\Pi_{k+1}^p = co-\Sigma_{k+1}^p. \quad (31)$$

Thus, $\Sigma_1^p = NP$, $\Pi_1^p = co-NP$, and Δ_2^p is the set of NP-easy problems. Further note that $\bigcup_{k \geq 0} \Delta_k^p = \bigcup_{k \geq 0} \Sigma_k^p = \bigcup_{k \geq 0} \Pi_k^p \subseteq PSPACE$.

The role of the “canonical” complete problem (w.r.t. polynomial transformability), which is played by SAT for Σ_1^p , is played by k -QBF for Σ_k^p . k -QBF is the problem of deciding whether the following quantified boolean formula is true:

$$\underbrace{\exists \vec{a} \forall \vec{b} \dots F(\vec{a}, \vec{b}, \dots)}_{k \text{ alternating quantifiers starting with } \exists}. \quad (32)$$

Turning now to the revision operations discussed in this paper, we first of all notice that the special belief revision problem of determining membership for a full meet revision, called FMR-problem, is comparably easy. With respect to Turing-reducibility, there is actually no difference to the complexity of ordinary propositional derivability, i.e., the FMR-problem is NP-equivalent.

Proposition 13 $FMR \in \Delta_2^p - (\Sigma_1^p \cup \Pi_1^p)$ provided $\Sigma_1^p \neq \Pi_1^p$.

Proof Sketch: If $\dot{+}$ is a full meet revision, $x \in Cn(Z) \dot{+} y$ can be solved by the following algorithm:

```
if  $Z \not\vdash \neg x$ 
  then  $Z \cup \{x\} \vdash y$ 
  else  $x \vdash y$ 
```

From this, membership in Δ_2^p follows.

Further, SAT can be polynomially transformed to FMR by solving $x \in Cn(x) \dot{+} T$, and unsatisfiability (SAT) can be polynomially transformed to FMR by solving $\perp \in Cn(\emptyset) \dot{+} x$. Hence, assuming $FMR \in NP \cup co-NP$ would lead to $NP = co-NP$. ■

The membership problem for simple base revision will be called SBR-problem. This problem is obviously more complicated than the FMR-problem. However, the added complexity is not overwhelming—from a theoretical point of view.

Theorem 14 SBR is Π_2^p -complete.

⁹The superscript p is only used to distinguish these sets from the analogous sets in the Kleene hierarchy.

Proof Sketch: We will prove that the complementary problem $Z \oplus x \not\vdash y$, which is called **SBR**, is Σ_2^P -complete. Hardness is shown by a polynomial transformation from 2-QBF to **SBR**. Let $\vec{a} = a_1, \dots, a_n$, let $\vec{b} = b_1, \dots, b_m$, and let $\exists \vec{a} \forall \vec{b} F(\vec{a}, \vec{b})$ be an instance of 2-QBF. Now set

$$Z = \{a_1, \dots, a_n, \neg a_1, \dots, \neg a_n, \neg F(\vec{a}, \vec{b})\}. \quad (33)$$

Now it is easy to see that

$$Z \oplus T \not\vdash \neg F(\vec{a}, \vec{b}) \text{ iff } \exists \vec{a} \forall \vec{b} F(\vec{a}, \vec{b}) \text{ is true.} \quad (34)$$

Membership in Σ_2^P follows from the following algorithm that needs nondeterministic polynomial time using an oracle for SAT:

1. Guess a set $Y \subseteq Z$.
2. Verify $Y \cup \{x\} \not\vdash x$.
3. Verify that there is no $z \in Z - Y$ such that $Y \cup \{z\} \not\vdash \neg x$.

This means that SBR is, on one hand, not much more difficult than FMR, and, on the other hand, apparently easier than derivability in the modal logics (e.g., K , T , and $S4$), which is a PSPACE-complete problem [Garey and Johnson, 1979, p. 262]. Asking for the computational significance of this result, the answer is somewhat unsatisfying. All problems in the polynomial hierarchy have the same property as the NP-complete problems, namely, that they can be solved in polynomial time if and only if $P = NP$. Further, all problems in the polynomial hierarchy can be solved by an exhaustive search that takes exponential time. However, from the structure of the algorithm used in the proof one sees that even if we restrict ourselves to polynomial methods for computing propositional satisfiability, there would still be the problem of determining the maximal consistent subsets Y .

Having now a very precise idea of the complexity of the SBR-problem, we may ask what the computational costs of introducing priorities are. In other words whether the membership problem for prioritized base revision, called **PBR**-problem, is more difficult than SBR.

Theorem 15 **PBR** is Π_2^P -complete.

Proof Sketch: Π_2^P -hardness is immediate by Theorem 14. Membership in Π_2^P follows from the fact that the algorithm used in the proof above can be easily adapted by guessing and verifying maximality for every priority class. ■

This means that we do not have to pay for introducing priority classes. In the case of default logics, the generalization from Poole's logic to RDT's does not increase the computational costs. Note also, that the

computational complexity of derivability for Brewka's LDT's is not easier because the reduction in the proof of Theorem 14 applies to the special case $\mathcal{F} = \emptyset$, as well.

The membership problem for unambiguous prioritized base revision, the **UBR**-problem, turns out to be easier than **SBR** and **PBR**.

Theorem 16 **UBR** $\in \Delta_2^P - (\Sigma_1^P \cup \Pi_1^P)$, provided $\Sigma_1^P \neq \Pi_1^P$.

Proof Sketch: The nondeterministic algorithm used in the proof above can be obviously modified so that it runs in deterministic polynomial time if the priority classes are singletons. Hence, **UBR** $\in \Delta_2^P$. Further, **UBR** $\notin \text{NP} \cup \text{co-NP}$ if $\text{NP} \neq \text{co-NP}$ follows from Proposition 13 because **FMR** is a special case of **UBR**. ■

From the proof, we can infer that if we can come up with a polynomial algorithm for satisfiability (by restricting the language, for instance), then unambiguous base revision will be itself polynomial. This result gives a formal justification for the claim made in [Nebel, 1989] that this form of revision is similar to the functionality the **RUP** system [McAllester, 1982] offers—in an abstract sense, though.¹⁰ The important point to note is that a feasible implementation of belief revision is possible if we restrict ourselves to polynomial methods for satisfiability by restricting the language or by tolerating incompleteness and by using a polynomial method for selecting among competing alternatives.

Finally, it may be interesting to compare syntax-based revision approaches with model-based approaches, such as the one proposed by Dalal [1988]. In order to do so, we first need some definitions. A *truth-assignment* \mathcal{I} is a function $\mathcal{I}: \Sigma \rightarrow \{\text{T}, \text{F}\}$. A *model* \mathcal{I} of a belief base Z is a truth-assignment that satisfies all propositions in Z in the classical sense, written $\models_{\mathcal{I}} Z$. *mod*(Z) denotes the set of all models of Z . $\delta(\mathcal{I}, \mathcal{J})$ denotes the number of propositional variables such that \mathcal{I} and \mathcal{J} map them to different truth-values. $g^m(\mathcal{M})$ is the set of truth assignments \mathcal{J} such that there is a truth-assignment $\mathcal{I} \in \mathcal{M}$ with $\delta(\mathcal{J}, \mathcal{I}) \leq m$. If Z is a finite belief base, then $G^m(Z)$ is some belief base such that $\text{mod}(G^m(Z)) = (g^m(\text{mod}(Z)))$. Although G^m is not a deterministic function, all possible results are obviously logically equivalent.

Now, model based revision, written $Z \circ x$ is defined

¹⁰The RUP systems provides the possibility to put premises into different likelihood classes. However, it seems to be the case that in resolving inconsistencies it could select non-maximal sets w.r.t. [McAllester, 1990, personal communication].

by:¹¹

$$Z \circ x \stackrel{\text{def}}{=} \begin{cases} G^m(Z) \cup \{x\} & \text{for the least } m \text{ s.t.} \\ & G^m(Z) \cup \{x\} \not\models \perp \\ \{x\} & \text{if } Z \models \perp \text{ or } x \models \perp. \end{cases} \quad (35)$$

Interestingly, the membership problem for model-based revision, called **MBR**-problem, has the same complexity as **UBR** and **FMR**. However, it is not obvious whether a restriction of the expressiveness of the logical language would lead to a polynomial algorithm in this case.

Theorem 17 $\mathbf{MBR} \in \Delta_2^p - (\Sigma_1^p \cup \Pi_1^p)$, provided $\Sigma_1^p \neq \Pi_1^p$.

Proof Sketch: Note that for any n , $G^n(Z) \not\models x$ is a problem that can be solved in nondeterministic polynomial time by guessing two truth assignment \mathcal{I}, \mathcal{J} and verifying in polynomial time that

1. $\models_{\mathcal{I}} Z$,
2. $\not\models_{\mathcal{J}} x$, and
3. $\delta(\mathcal{I}, \mathcal{J}) \leq n$.

Further, solving $G^n(Z) \cup \{x\} \not\models y$ for fixed n also requires only nondeterministic polynomial time. Hence, $\mathbf{MBR} \in \Delta_2^p$. Since SAT and $\overline{\text{SAT}}$ can be polynomially transformed to **MBR**, $\mathbf{MBR} \in \text{NP} \cup \text{co-NP}$ only if $\text{NP} = \text{co-NP}$. ■

Reconsidering the complexity results, there appears to be an interesting pattern. Note that the best result for a belief revision problem we can hope for is membership in Δ_2^p because the problem involves consistency and inconsistency problems. While, the revision operations satisfying all the Gärdenfors postulates, namely, **FMR**, **UBR**, and **MBR** (for the latter see [Dalal, 1988]) turn out to be in this class, the revision operations that do not satisfy (+8) cannot be shown to be in this class. An interesting question is what conditions are in fact responsible for membership in Δ_2^p .

8 SUMMARY AND OUTLOOK

Syntax-based approaches to belief revision, the class of partial meet revisions based on epistemic relevance, and ranked default theories (RDT's)—a generalization of Poole's and Brewka's approaches to default reasoning—turn out to be strictly equivalent in the case of finitary propositional logic. One of the consequences is that RDT's do not satisfy rational monotony w.r.t. the set of hard facts. Further, we are able to apply the complexity results for belief revision directly to reasoning in RDT's.

¹¹This definition is a slight extension of the definition given in [Dalal, 1988] that takes also care of the limiting cases when Z or x is inconsistent.

The complexity results for revision and default reasoning confirm the intuition that unambiguous prioritized base revision is not harder but apparently less complex than general prioritized base revision [Doyle, 1990, Sect. 3.2], which in turn is not harder than simple base revision. An interesting point is that model-based revision as proposed by Dalal is still NP-easy.

One of the open questions is, whether the correspondence between belief revision and nonmonotonic logic holds for the infinite case as well. However, for this purpose the theory of epistemic change has to be extended so that belief sets cannot only be revised by sentences but also by other belief sets. Another interesting question in this context is whether there is a natural condition on belief revision and nonmonotonic consequence operations that characterizes syntax-based approaches completely.

Finally, the observation that all “fully rational” revision operations analyzed in this paper share the property of being NP-easy suggests to analyze this class of revision operations in more detail in order to detect interesting tractable special cases.

Acknowledgement

I would like to thank Gerd Brewka, Jon Doyle, David Makinson, Peter Gärdenfors, and David McAllester for discussions about the subject of this paper and Gerd and David Makinson for comments on an earlier draft.

This work was supported by the German Ministry for Research and Technology BMFT under contract ITW 8901 8 as part of the WIP project.

References

- [Alchourrón *et al.*, 1985] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2): 510–530, June 1985.
- [Brachman, 1990] Ronald J. Brachman. The future of knowledge representation. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 1082–1092, Boston, Mass., August 1990.
- [Brewka, 1989] Gerhard Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1043–1048, Detroit, Mich., August 1989.
- [Brewka, 1990] Gerhard Brewka. *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge University Press, Cambridge, England, 1990. To appear.
- [Dalal, 1988] Mukesh Dalal. Investigations into a theory of knowledge base revision: Preliminary report.

- [In Proceedings of the 7th National Conference of the American Association for Artificial Intelligence, pages 475–479, Saint Paul, Minn., August 1988.]
- [Doyle, 1990] Jon Doyle. Rational belief revision. Presented at the Third International Workshop on Non-monotonic Reasoning, Stanford Sierra Camp, Cal., June 1990.
- [Fagin et al., 1983] Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the semantics of updates in databases. In *2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 352–365, Atlanta, Ga., 1983.
- [Gärdenfors and Makinson, 1988] Peter Gärdenfors and David Makinson. Revision of knowledge systems using epistemic entrenchment. In M. Vardi, editor, *Proceedings of the 2nd Workshop on Theoretical Aspects of Reasoning about Knowledge*. Morgan Kaufmann, Los Altos, Cal., 1988.
- [Gärdenfors, 1988] Peter Gärdenfors. *Knowledge in Flux—Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge, Mass., 1988.
- [Gärdenfors, 1990] Peter Gärdenfors. Belief revision and nonmonotonic logic: Two sides of the same coin? In L. C. Aiello, editor, *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 768–773, Stockholm, Sweden, August 1990.
- [Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, Cal., 1979.
- [Ginsberg and Smith, 1987] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. In F. M. Brown, editor, *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*, pages 233–258. Morgan Kaufmann, Los Altos, Cal., 1987.
- [Ginsberg, 1986] Matthew L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30(1):35–79, October 1986.
- [Katsuno and Mendelzon, 1989] Hirofumi Katsuno and Alberto O. Mendelzon. A unified view of propositional knowledge base updates. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 1413–1419, Detroit, Mich., August 1989.
- [Katsuno and Mendelzon, 1990] Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. Technical Report KRR-TR-90-6, University of Toronto, Computer Science Department, Toronto, Ont., August 1990.
- [Makinson and Gärdenfors, 1990] David Makinson and Peter Gärdenfors. Relations between the logic of theory change and nonmonotonic logic.
- [In A. Fuhrmann and M. Morreau, editors, *Proceedings of the Konstanz Workshop on Belief Revision*. Springer-Verlag, Berlin, Germany, 1990. To appear.]
- [McAllester, 1982] David A. McAllester. Reasoning utility package user's manual. AI Memo 667, AI Laboratory, Massachusetts Institute of Technology, Cambridge, Mass., April 1982.
- [Myers and Smith, 1988] Karen L. Myers and David E. Smith. The persistence of derived information. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence*, pages 496–500, Saint Paul, Minn., August 1988.
- [Nebel, 1989] Bernhard Nebel. A knowledge level analysis of belief revision. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 301–311, Toronto, Ont., May 1989.
- [Nebel, 1990] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 1990.
- [Nebel, 1991] Bernhard Nebel. Belief revision and default reasoning: syntax-based approaches. DFKI Report, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, West Germany, 1991. To appear.
- [Newell, 1982] Allen Newell. The knowledge level. *Artificial Intelligence*, 18(1):87–127, 1982.
- [Poole, 1988] David Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.
- [Rao and Foo, 1989] Anand S. Rao and Norman Y. Foo. Formal theories of belief revision. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 369–380, Toronto, Ont., May 1989.
- [Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1): 81–132, April 1980.
- [Winslett, 1988] Marianne S. Winslett. Reasoning about action using a possible models approach. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence*, pages 89–93, Saint Paul, Minn., August 1988.
- [Winslett, 1989] Marianne S. Winslett. Sometimes updates are circumscription. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 859–863, Detroit, Mich., 1989.

Generalizing Atoms in Constraint Logic

C. David Page Jr. and Alan M. Frisch
 Dept. of Computer Science and Beckman Institute
 University of Illinois
 405 North Mathews Ave.
 Urbana, IL 61801

Abstract

This paper studies the generalization of atomic formulas, or atoms, that are augmented with constraints on or among their terms. The atoms may also be viewed as definite clauses whose antecedents express the constraints. Atoms are generalized relative to a body of background information about the constraints. The paper first examines generalization of atoms with only monadic constraints. The paper develops an algorithm for the generalization task and discusses algorithm complexity. It then extends the algorithm to apply to atoms with constraints of arbitrary arity. The paper also presents semantic properties of the generalizations computed by the algorithms, making the algorithms applicable to such problems as abduction, induction, and knowledge base vivification. The paper emphasizes the application to induction and presents a pac-learning result for constrained atoms.

1 Introduction

Unification plays an important role in many automated deduction systems. It comes in a number of forms including term unification, string unification, unification with built-in equality, sorted unification, and feature structure unification, as well as variations of these. The common feature of all these forms of unification is that they compute maximal lower bounds in some partially-ordered set of syntactic objects. Turning unification on its head yields an operation called "generalization," or "anti-unification," that computes minimal upper bounds. As with unification, one can imagine many forms of generalization. We conjecture that generalization will play a role in non-deductive reasoning (e.g., abduction, induction and analogy) as important as that played by unification in deductive reasoning.

Section 2 of the paper reviews ordinary generalization, which has been studied by Plotkin [1970; 1971], Reynolds [1970], Lassez, Maher and Marriott [1988], and Lassez and Marriott [1987]. Ordinary generalization is a purely syntactic operation that operates solely on the basis of expression structure. Section 2 also reviews sorted generalization [Frisch and Page, 1990], an operation that acts on atoms whose variables are constrained by associated sorts, or monadic constraints. Sorted generalization is governed by the logical consequences of a taxonomic theory that describes the sorts. Section 3 first extends the generalization operation to allow monadic constraints on arbitrary terms in the atoms, rather than on variables only. Section 3 extends the generalization operation further still, so that it operates on atoms with constraints of arbitrary arity among their terms. The resulting operation acts with respect to a theory that describes the constraints.

Elsewhere [Frisch and Page, 1990] we examine applications of sorted generalization to abduction, induction, knowledge base vivification, and analogical reasoning. These applications are based on semantic properties of sorted generalization. Section 4 of this paper shows that constraint generalization has analogous semantic properties. Section 4 then concentrates on the application to induction, discussing polynomial-time pac-learning of constrained atoms and polynomial-time learning of constrained atoms by equivalence queries.

From a constraint logic programming point of view, an atomic formula, A , with attached constraints can be seen as a definite clause whose head is A and whose body contains zero or more constraint literals but no ordinary literals. Alternatively, such a formula can be seen as a definite clause of ordinary logic whose head is A and whose body is a conjunction of all the constraints. Thus, our work is related to Buntine's [1988] work on generalized subsumption. Section 5 examines this relationship.

2 Background

The most general common instance, e , of a set, E , of atoms may be found with unification and characterizes the common instances of E : an atom is a common instance of E if and only if it is an instance of e . Likewise, the most specific generalization (*msg*), g , of E characterizes the atoms that are more general than every atom in E : an atom is more general than every member of E if and only if it is more general than g . The ordinary generalization operation computes this most specific generalization (*msg*) of a set of atoms.

Algorithms for ordinary generalization have appeared elsewhere [Plotkin, 1970; Plotkin, 1971; Reynolds, 1970; Lassez *et al.*, 1988]. The following example illustrates the interesting aspects of ordinary generalization.

Example 1

Let $E = \{\text{loves}(\text{clyde}, \text{mom}(\text{clyde})), \text{loves}(\text{jumbo}, \text{mom}(\text{jumbo}))\}$.

Then $\text{loves}(x, \text{mom}(x))$ is an *msg* of E .

Notice that the common structure of the atoms in E is preserved in the *msg* of E . Notice also that the variable x repeats in the *msg* the same way that *clyde* and *jumbo* repeat in the first and second atoms, respectively, of E . The *msg* of a set of atoms whose sizes (number of variable and function symbol occurrences) sum to S can be computed in time $O(S)$.

Other researchers (for example, [Mitchell *et al.*, 1983]) have studied generalization with respect to *taxonomic background information*. Whereas generalizing “Clyde is gray” and “Jumbo is gray” yields the more general assertion, “Everything is gray,” generalization with respect to the background information that Clyde and Jumbo are elephants yields the more specific result, “Elephants are gray.” Sorted generalization combines ordinary generalization with the use of taxonomic background information. An important distinction between sorted generalization and such earlier work is that sorted generalization operates with respect to taxonomic theories of a very general form: first-order theories in which sorts act as monadic predicates. One result of such general taxonomic theories is that some sets of atoms have no most specific generalization. Rather, they may have a set of *minimal* generalizations, none more specific than another, and these minimal generalizations may differ significantly. Examples 3 and 4 in this section illustrate this aspect of sorted generalization. Most of the following material in this section repeats earlier work [Frisch and Page, 1990] that is needed as background for the remainder of this paper.

Sorted generalization operates on sorted atoms, which are like ordinary atoms except that they may contain *sorted variables*, variables that have *sort symbols* at-

tached. (We also say that sorted terms are terms that may contain sorted variables.) The sorted variable $x:\text{ELEPHANT}$, for example, is restricted to range over the set of all elephants. Sorted generalization operates with respect to a taxonomic theory, or sort theory, that describes the sorts. A sort theory treats the sorts as monadic predicates. Several examples in the paper use the following sort theory:

$$\Sigma_1 = \{\forall x \text{UNIV}(x), \text{GRAY}(\text{mom}(\text{clyde})), \\ \text{GRAY}(\text{mom}(\text{jumbo})), \\ \text{ELEPHANT}(\text{clyde}), \text{IN-CIRCUS}(\text{clyde}), \\ \text{ELEPHANT}(\text{jumbo}), \text{IN-CIRCUS}(\text{jumbo}), \\ \forall x \text{ ELEPHANT}(x) \rightarrow \text{ELEPHANT}(\text{mom}(x)), \\ \forall x \text{ ELEPHANT}(x) \rightarrow \text{MAMMAL}(x)\}.$$

We are often interested in whether one sort is a subset of another sort according to a given sort theory Σ . τ' is a subsort of τ according to Σ if and only if $\Sigma \models \forall x(\tau'(x) \rightarrow \tau(x))$; we write $\tau' \preceq_{\Sigma} \tau$. If $\tau' \preceq_{\Sigma} \tau$ and $\tau \not\preceq_{\Sigma} \tau'$, then we write $\tau' <_{\Sigma} \tau$.

Where Σ is a sort theory, we say that one sorted atom is Σ -more general (\geq_{Σ}) than a second if and only if some substitution θ maps the first to the second, and θ respects the sorts of the variables. A substitution θ respects the sorts of variables relative to a sort theory Σ if and only if for every variable $x:\tau$, $(x:\tau)\theta$ is a term t such that $\Sigma \models \bar{\forall}t(t)$.¹ If $e \geq_{\Sigma} e_1$ and $e \geq_{\Sigma} e_2$, we also say that e Σ -subsumes e_1 and e_2 and that e is a Σ -generalization of e_1 and e_2 . If $e_1 \geq_{\Sigma} e_2$ and $e_2 \not\geq_{\Sigma} e_1$, we also write $e_1 >_{\Sigma} e_2$. The \geq_{Σ} relation is particularly useful because under natural definitions for the universal and existential closures of sorted atoms [Frisch and Page, 1990], for any sorted atomic formulas, α_1 and α_2 , and any sort theory, Σ : $\alpha_1 \geq_{\Sigma} \alpha_2$ if and only if $\{\forall \alpha_1\} \cup \Sigma \models \bar{\forall} \alpha_2$ if and only if $\{\exists \alpha_2\} \cup \Sigma \models \bar{\exists} \alpha_1$.

Example 2

$\text{eats}(x:\text{ELEPHANT}, \text{nuts}) \geq_{\Sigma_1} \text{eats}(\text{clyde}, \text{nuts})$.

To see this notice that for $\theta_1 = \{\text{clyde}/x:\text{ELEPHANT}\}$,

$$\text{eats}(x:\text{ELEPHANT}, \text{nuts})\theta = \text{eats}(\text{clyde}, \text{nuts}).$$

Notice further that Σ_1 entails that *clyde* is an elephant. On the other hand,

$$\text{eats}(x:\text{ELEPHANT}, \text{nuts}) \not\geq_{\Sigma_1} \text{eats}(\text{fido}, \text{nuts}).$$

To see this, notice that any substitution mapping $\text{eats}(x:\text{ELEPHANT}, \text{nuts})$ to $\text{eats}(\text{fido}, \text{nuts})$ must map $x:\text{ELEPHANT}$ to *fido*. But Σ_1 does not entail that *fido* is an elephant.

Example 3

Let $E = \{\text{eats}(\text{clyde}, \text{nuts}), \text{eats}(\text{jumbo}, \text{nuts})\}$.

¹For any expression ψ , $\bar{\forall}\psi$ denotes the universal closure of ψ , the result of universally quantifying all free variables of ψ . Similarly, $\bar{\exists}\psi$ denotes the existential closure of ψ .

E has two maximally-specific sorted generalizations with respect to Σ_1 :

$\text{eats}(x:\text{ELEPHANT}, \text{nuts})$, and
 $\text{eats}(y:\text{IN-CIRCUS}, \text{nuts})$.

E has a most specific unsorted generalization:

$\text{eats}(x, \text{nuts})$.

We would like to use a Σ -generalization to characterize the sorted atoms that are Σ -more general than any given set of sorted atoms. But Example 3 shows that there may be no most specific Σ -generalization: neither $\text{eats}(x:\text{ELEPHANT}, \text{nuts})$ nor $\text{eats}(y:\text{IN-CIRCUS}, \text{nuts})$ is more specific, with respect to Σ , than the other. An analogous characterization therefore requires a set of maximally specific Σ -generalizations. Specifically, the goal of sorted generalization is to find a *complete set of incomparable Σ -generalizations* (or CIG_Σ) for a given set of sorted atoms. Σ -generalizations are incomparable if none is Σ -more general than another. A set G of maximally specific, incomparable Σ -generalizations of a set E of sorted atoms is a CIG_Σ of E if and only if any other Σ -generalization of E is Σ -more general than some sorted atom in G . Every set (even infinite) of sorted atoms has a finite CIG_Σ . For some sets the CIG_Σ is empty.

The following example illustrates an additional distinction between sorted and ordinary generalization, which a sorted generalization algorithm must accommodate. In Example 4 “mom” is a function symbol, so E is a set of terms.

Example 4

Let $E = \{\text{mom}(\text{clyde}), \text{mom}(\text{jumbo})\}$.

Then a CIG_{Σ_1} of E is

$\{y:\text{GRAY}, \text{mom}(x:\text{ELEPHANT}), \text{mom}(z:\text{IN-CIRCUS})\}$,

and $\text{mom}(x)$ is a most specific unsorted generalization.

In Example 4, $y:\text{GRAY}$ Σ_1 -subsumes neither $\text{mom}(x:\text{ELEPHANT})$ nor $\text{mom}(z:\text{IN-CIRCUS})$. This contrasts with the unsorted case, where a variable always subsumes any other term. We refer to a variable that Σ -subsumes all members of a set of (possibly sorted) terms as a *variable generalization* of that set. We refer to a non-variable sorted atom that Σ -subsumes all members of a set as a *structured generalization*. A sorted generalization algorithm cannot assume that a variable generalization of a set of (sorted) terms Σ -subsumes every structured generalization of that set. It must therefore compare the variable generalizations with the structured generalizations.

Because variable generalizations need not Σ -subsume structured generalizations, members of a CIG_Σ may have different structures. In the CIG_{Σ_1} of Example 4, one generalization is a variable while the other

two are built from the function symbol “mom.” Such structural differences can become more remarkable as the expressions being generalized grow more complex. That variable generalizations need not Σ -subsume structured generalizations is a result of *function polymorphism*. A function is polymorphic if terms built from that function’s symbol may denote members of different sorts based on the arguments in the terms. *mom* is a polymorphic function, because whether a term built from “mom” denotes a member of the sort GRAY depends on the argument to *mom*. If the argument is *clyde* or *jumbo*, the term denotes something gray; otherwise it does not. If *mom* were instead a monomorphic function, that is, if all *moms* were known to be gray, the variable $y:\text{GRAY}$ would Σ_1 -subsume any structured generalization.

Elsewhere [Frisch and Page, 1990] we present an algorithm that computes sorted generalization given oracles for two kinds of *taxonomic queries*. These two kinds of queries are actually equivalent to a single kind of taxonomic query that, given a sort theory, Σ , and a set of sorted terms, $\{t_1, \dots, t_n\}$, asks for the minimal (under the \preceq_Σ ordering) sorts in the set $\{\tau \mid \Sigma \models \bar{\forall} \tau(t_i) \text{ for all } 1 \leq i \leq n\}$. Therefore an oracle, or decision procedure, for these taxonomic queries is sufficient for computation of sorted generalization. An oracle for these queries is in fact necessary as well. In general such an oracle cannot be computed, but restrictions can be placed on the sort theory so that it can be. Unfortunately, the size of a CIG_Σ can be exponential in the sizes of the sorted atoms generalized, so generalization can require exponential time and space. Though we can represent the CIG_Σ compactly, in polynomial size, computing even a compactly represented CIG_Σ is *NP-hard*.

We close this section with a new result. Probably the most obvious application of generalization is inductive learning. Two widely-appreciated models of successful inductive learning are *polynomial pac-learning* [Valiant, 1984; Blumer et al., 1989] and *polynomial learning by equivalence queries* [Angluin, 1988a; Angluin, 1988b]. Learning in the second model implies learning in the first [Angluin, 1988b]. Can sorted atoms be learned in these models given efficient response to taxonomic queries? Before answering this question, we first define these inductive learning models as they apply to sorted atoms.²

For any two sorted atoms, α_1 and α_2 , and any sort theory, Σ , α_1 is said to *cover* α_2 with respect to Σ if and only if $\alpha_1 \geq_\Sigma \alpha_2$. Two sorted atoms are equivalent with respect to Σ if and only if they cover the same sorted atoms with respect to Σ . A sorted atom α_2 is labeled as a *positive example* by another sorted atom, α_1 , and a sort theory, Σ , if α_1 covers α_2 with respect

²The definitions are based on definitions given in [Angluin, 1988a; Angluin, 1988b; Blumer et al., 1989].

to Σ . Otherwise, α_2 is labeled as a *negative example* by α_1 and Σ . Each sorted atom has an associated size, which is the number of variable and function symbol occurrences in the atom.

Definition 5 An equivalence query asks whether a particular sorted atom (the predicted atom) is equivalent to an unknown target sorted atom with respect to the sort theory Σ . An oracle for equivalence queries answers yes to any given query if the predicted atom is equivalent to the target with respect to Σ ; it answers no otherwise. The oracle gives a counterexample if the predicted atom is not equivalent to the target. A counterexample to an equivalence query is any sorted atom labeled differently by the predicted and target concepts with respect to Σ .

Definition 6 An algorithm A is a polynomial-time learning algorithm for sorted atoms with fixed Σ if and only if for any fixed sort theory Σ , for which any taxonomic query is answered in time polynomial in the size of the query, there exists a polynomial $p(n, m)$ such that: for any target sorted atomic formula α , when A is run with an oracle to answer equivalence queries for α with respect to Σ , A halts and outputs a sorted atom that is equivalent to α . Moreover, at any point during the run, the time used by A to that point is bounded by $p(n, m)$, where n is the size of α , and m is the length of the longest counterexample seen to that point. Sorted atoms, with fixed Σ , are polynomially learnable by equivalence queries if and only if there exists such an algorithm A .

Definition 7 An algorithm A is a polynomial-time pac-learning algorithm for sorted atoms with fixed Σ if and only if for any fixed sort theory, Σ , for which the response to any taxonomic query is returned in time polynomial in the size of the query, A has the following property. For all $0 < \epsilon, \delta < 1$, and $n \geq 1$ there exists a sample size $m(\epsilon, \delta, n)$, polynomial in $1/\epsilon$, $1/\delta$, and n , such that for all sorted atoms, α , of size n or less, and all probability distributions, P , on the sorted atoms, given a random sample of $m(\epsilon, \delta, n)$ sorted atoms drawn independently according to P and labeled according to α and Σ , A produces, with probability at least $1 - \delta$, a sorted atom, α' , such that the probability that a given sorted atom drawn according to P is labeled differently, with Σ , by α and α' is at most ϵ . Sorted atoms are polynomially pac-learnable for fixed Σ if and only if such an algorithm A exists.

Applying one result of Computational Learning Theory [Pitt and Valiant, 1988], we find that learning sorted atoms with respect to a given sort theory, Σ , in these models is possible only if a (possibly randomized, for pac-learning) polynomial-time algorithm for the following problem exists. Given a set of sorted atoms, each labeled as either a positive or negative example, determine whether there exists a sorted atom,

α , that covers, with Σ , all of the positive examples but none of the negative examples. Such a sorted atom is said to be *consistent with the examples*, and we call the problem of determining whether such a sorted atom exists the *consistency problem* for sorted atoms. We have noted that sorted generalization requires exponential time, and that even with a compact representation it requires exponential time assuming $P \neq NP$. The following result tells us that this complexity carries over to the consistency problem, and so to both pac-learning and learning by equivalence queries if we assume $R \neq NP$.³

Theorem 8 Sort theories Σ exist for which the consistency problem is NP -hard, even though any taxonomic query about Σ is answered in time polynomial in the size of the query.

Therefore, assuming $R \neq NP$, no algorithms exist that polynomially pac-learn sorted atoms or polynomially learn sorted atoms by equivalence queries.

3 A More General Language

In some cases generalization and inductive learning become easier if we broaden the language of acceptable generalizations. In this section we extend the language of both observations and acceptable generalizations. This extension yields efficient generalization and even positive results (provided in Section 4) for polynomial pac-learning and polynomial learning by equivalence queries. We extend the language in stages, eventually replacing sorts, which may be viewed as monadic constraints on variables, with arbitrary constraints among terms. Generalization with constraint information may be viewed as a dual⁴ of constraint unification, a form of unification that is useful in constraint deduction and constraint logic programming.

3.1 Extended Sorted Generalization

We begin our extension of the language by changing the syntax of sorted atoms. Following work in constraint logic, we use separate specifications for the structure of a sorted atom and the sorts, or constraints, on its variables. In the new syntax, for example, the sorted atom

$$\text{loves}(x:\text{ELEPHANT}, \text{mom}(z:\text{IN-CIRCUS}))$$

is written as

$$\text{loves}(x, z)/\text{ELEPHANT}(x) \wedge \text{IN-CIRCUS}(z).$$

The structure of the sorted atom appears first, followed by a slash and then the specifications of the sorts. The

³That is, if we assume the NP -complete problems cannot be solved in random polynomial time.

⁴We use "dual" in a non-technical sense.

expression that represents the structure of the sorted atom is called the *head* of the sorted atom. The conjunction of sort specifications is called the *constraint* of the sorted atom. Sorted terms are like sorted atoms except the head is a term.

Having changed the syntax in this way, we may constrain any terms in a sorted atom, rather than the variables only, to belong to a given sort. Thus sorted atoms have not only sorted variables, but sorted terms in general. For example, we may now constrain $\text{mom}(z)$ to be a lawyer. Thus

$$\text{loves}(x, \text{mom}(z)) / \text{ELEPHANT}(x) \wedge \text{IN-CIRCUS}(z) \wedge \text{LAWYER}(\text{mom}(z))$$

is a legal sorted atom. It says “ x , which is an elephant, loves the mom of z , where z is in a circus and the mom of z is a lawyer.” Note that we require all terms that appear in the constraint of a sorted atom to appear in the head as well. Thus, for example,

$$\text{loves}(x, z) / \text{LAWYER}(\text{father}(z))$$

is not a legal sorted atom. Absence of this requirement leads to greatly increased computational costs in generalization.

We further expand the definition of sorted atoms to allow terms to be constrained with multiple sorts. For example,

$$\text{loves}(x, z) / \text{ELEPHANT}(x) \wedge \text{ELEPHANT}(z) \wedge \text{IN-CIRCUS}(z)$$

becomes a sorted atom. It says “ x , which is an elephant, loves z , which is an elephant and a circus animal.”

Section 2 takes the size of a sorted atom to be the number of variable and function symbol occurrences it contains. Now that we have more general constraints on terms, it is useful to distinguish between the *head size* of a sorted atom, which is the number of variable and function symbol occurrences in the head, and the *total size* of a sorted atom, which is the head size plus the number of variable, function symbol, and constraint predicate occurrences in the constraint of the sorted atom.

Having expanded the definition of sorted atoms, we must expand our semantic rules for quantification. Let $\alpha = A/a_1 \wedge \dots \wedge a_n$ be a sorted atom. Because we are interested in only the universal and existential closures of sorted atoms, it is sufficient to specify that $\bar{\forall}\alpha$ is equivalent to $\bar{\forall}(a_1 \wedge \dots \wedge a_n \rightarrow A)$ and $\exists\alpha$ is equivalent to $\exists(a_1 \wedge \dots \wedge a_n \wedge A)$. Note that a universally-closed sorted atom is equivalent to a definite clause whose body (the constraint) is built of sort symbols and terms that appear in its head.

We would like to again base our generalization algorithm on a Σ -more general (\geq_Σ) relation. Recall

that under the original definition of sorted atoms, one sorted atom is Σ -more general than a second if some substitution maps the first to the second and respects the sorts of all variables with respect to Σ . Recall also that a substitution, θ , respects the sorts of all variables if for every variable $x:\tau$, θ maps $x:\tau$ to a term t for which $\Sigma \models \bar{\forall}\tau(t)$. But under the new definition of sorted atoms, arbitrary terms, rather than variables only, may be assigned sorts. Therefore, insisting only that a substitution must respect the sorts of the variables is too weak. We instead want to require substitutions that respect the sorts of all terms.

Our new syntax for sorted atoms lets us phrase the definition of \geq_Σ another way. Let $\alpha = A/a_1 \wedge \dots \wedge a_n$ and $\alpha' = A'/a'_1 \wedge \dots \wedge a'_m$ be sorted atoms. We say that $\alpha \geq_\Sigma \alpha'$ if a substitution θ exists for which $A\theta = A'$ and $\Sigma \models \bar{\forall}(a'_1 \wedge \dots \wedge a'_m \rightarrow (a_1 \wedge \dots \wedge a_n)\theta)$. In the cases where α and α' are sorted atoms under the original definition of sorted atoms—that is, where sorts are specified on only variables in the atoms, and at most one sort is specified on each variable—this definition of \geq_Σ is equivalent to the original one. Note that this relation may be applied to sorted terms as well.

Example 9

$$\begin{aligned} & \text{loves}(x, y) / \text{MAMMAL}(x) \wedge \text{MAMMAL}(y) \\ & \geq_{\Sigma_1} \text{loves}(z, \text{mom}(z)) / \text{ELEPHANT}(z). \end{aligned}$$

To see this, notice first that for any substitution θ that maps x to z and y to $\text{mom}(z)$:

$$\text{loves}(x, y)\theta = \text{loves}(z, \text{mom}(z)).$$

Second, note that

$$\begin{aligned} & (\text{MAMMAL}(x) \wedge \text{MAMMAL}(y))\theta \\ & = \text{MAMMAL}(z) \wedge \text{MAMMAL}(\text{mom}(z)). \end{aligned}$$

Because Σ_1 entails that the mom of an elephant is an elephant and that all elephants are mammals,

$$\Sigma_1 \models \bar{\forall}(\text{ELEPHANT}(z) \rightarrow (\text{MAMMAL}(z) \wedge \text{MAMMAL}(\text{mom}(z))).$$

The following is an example involving an expression that is a sorted atom under the new definition but not under the old one.

Example 10

$$\begin{aligned} & \text{loves}(x, \text{mom}(x)) / \text{ELEPHANT}(x) \wedge \text{IN-CIRCUS}(x) \wedge \\ & \quad \text{GRAY}(\text{mom}(x)) \\ & \geq_{\Sigma_1} \text{loves}(\text{clyde}, \text{mom}(\text{clyde})). \end{aligned}$$

For any substitution θ that maps x to clyde ,

$$\text{loves}(x, \text{mom}(x))\theta = \text{loves}(\text{clyde}, \text{mom}(\text{clyde})).$$

Note further that

$$\begin{aligned}
 & (\text{ELEPHANT}(x) \wedge \text{IN-CIRCUS}(x)) \\
 & \quad \wedge \text{GRAY}(\text{mom}(x)))\theta \\
 = & \text{ELEPHANT}(\text{clyde}) \wedge \text{IN-CIRCUS}(\text{clyde}) \\
 & \quad \wedge \text{GRAY}(\text{mom}(\text{clyde})).
 \end{aligned}$$

We can easily verify that

$$\begin{aligned}
 \Sigma_1 \models & \text{ELEPHANT}(\text{clyde}) \wedge \text{IN-CIRCUS}(\text{clyde}) \\
 & \wedge \text{GRAY}(\text{mom}(\text{clyde})).
 \end{aligned}$$

Finally, as a negative example of the \geq_{Σ} relation,

Example 11

$$\begin{aligned}
 & \text{loves}(x, \text{mom}(x)) / \text{ELEPHANT}(x) \wedge \text{IN-CIRCUS}(x) \\
 \not\models_{\Sigma_1} & \text{loves}(x, \text{mom}(x)) / \text{MAMMAL}(x) \wedge \text{GRAY}(\text{mom}(x)).
 \end{aligned}$$

The relation fails to hold because

$$\begin{aligned}
 \Sigma_1 \not\models & \forall((\text{MAMMAL}(x) \wedge \text{GRAY}(\text{mom}(x))) \\
 & \rightarrow (\text{ELEPHANT}(x) \wedge \text{IN-CIRCUS}(x))).
 \end{aligned}$$

Having expanded the definitions of sorted atoms and the \geq_{Σ} relation, any set of sorted atoms has a singleton CIG_{Σ} . That is, any set of sorted atoms has a most specific Σ -generalization (msg_{Σ}). To see why, let's rework Example 4 using the extended definitions of sorted atoms and sorted terms.

Example 12

Let E be $\{\text{mom}(\text{clyde}), \text{mom}(\text{jumbo})\}$.

Then an msg_{Σ_1} of E is

$$\begin{aligned}
 & \text{mom}(x) / \text{ELEPHANT}(x) \wedge \text{IN-CIRCUS}(x) \wedge \\
 & \quad \text{GRAY}(\text{mom}(x)).
 \end{aligned}$$

The CIG_{Σ_1} in Example 4 has multiple generalizations for two reasons. The first reason is that two minimal sorts, rather than one, contain both *clyde* and *jumbo*. Therefore, under the original definition of sorted atoms, we obtain two minimal generalizations of *clyde* and *jumbo*— $x:\text{ELEPHANT}$ and $x:\text{IN-CIRCUS}$ —which lead to two minimal generalizations of *mom(clyde)* and *mom(jumbo)*. Under the new definition of sorted atoms, we can attach multiple sorts to a variable, thereby obtaining one minimal generalization of *clyde* and *jumbo*—a variable, x , belonging to both *ELEPHANT* and *IN-CIRCUS* sorts. Notice that both sorts are attached to x in Example 12. The second reason for multiple minimal generalizations in Example 4 is that both *mom(clyde)* and *mom(jumbo)* belong to the sort *GRAY*, while neither *mom(x:ELEPHANT)* nor *mom(x:IN-CIRCUS)* belong to that sort. Therefore, a variable of the sort *GRAY* is a third minimal generalization. Under the new definition of sorted atoms, we can always add sorts on an entire non-variable term to make it more specific, with respect to the sort theory, than any given variable. We do so in Example 12 by constraining *mom(x)* to belong to the sort *GRAY*, thus making it more specific than a variable of sort *GRAY*.

The Extended Sorted Generalization (Extended SG) Algorithm (Figure 1) computes the msg_{Σ} of any finite

set of extended sorted atoms. The presentation of the algorithm requires one additional definition. Given an expression e_1 and a particular occurrence γ of a term in e_1 , the Extended SG Algorithm needs to be able to select from another expression e_2 the term t that occurs at the same position in e_2 as does γ in e_1 . We say that t is the term in e_2 corresponding to γ in e_1 . For example, if e_1 is $P(f(a), a)$ and e_2 is $P(f(x), f(x))$, then x is the term in e_2 corresponding to the first occurrence of a in e_1 , and $f(x)$ is the term in e_2 corresponding to the second occurrence of a in e_1 . More precisely we define the term in e_2 corresponding to γ in e_1 to be t if, and only if:

- γ is e_1 and t is e_2 , or
- e_1 is of the form $f(s_1, \dots, s_n)$ and e_2 is of the form $f(u_1, \dots, u_n)$, where f is a function symbol or predicate, and γ is in s_i and t is the term in u_i corresponding to γ in s_i .

If one occurrence of a term t_1 in an expression e corresponds to a term t_2 in an expression e' , then so does every other occurrence of t_1 in e . In this case we sometimes speak of the term in e' that corresponds to a given term, rather than term occurrence, in e .

The Extended SG Algorithm interacts with a sort theory Σ only through the following oracle. For all $1 \leq i \leq n$, let $a_{i,1} \wedge \dots \wedge a_{i,m_i}$ be any constraint and let t_i be any term. The oracle returns the members of the set $\{\tau \mid \Sigma \models \forall(a_{i,1} \wedge \dots \wedge a_{i,m_i} \rightarrow \tau(t_i))$ for all $1 \leq i \leq n\}$ that are minimal under the \leq_{Σ} ordering. This oracle is stronger than the analogous one used in sorted generalization as described in Section 2. Another way of specifying the oracle of Section 2 is that it is a restriction of the oracle used by the Extended SG Algorithm to cases where each constraint specifies sorts for only variables and specifies at most one sort for each variable. The oracle used by the Extended SG Algorithm is in fact necessary for sorted generalization in the extended language if we wish to avoid constraints with redundancy.⁵ given any algorithm for extended sorted generalization that avoids redundancy in the constraints, it can be used to compute the oracle. An obvious, weaker oracle is necessary and sufficient if we allow redundancy in constraints.

Theorem 13 Let Σ be a sort theory such that any taxonomic query of size s (s variable, function symbol, and constraint predicate occurrences) about Σ is made and answered in time at most $Q(s)$.⁶ The Extended Sorted Generalization Algorithm computes an msg_{Σ} of

⁵ A constraint contains redundancy if it contains sort specifications $\tau_1(t)$ and $\tau_2(t)$, where $\tau_1 \preceq_{\Sigma} \tau_2$.

⁶ For some sort theories, it may be desirable to measure the time complexities of queries in a more detailed way, such as by dividing the size of a query into the sizes of the terms and the sizes of the constraints used in the query.

Extended Sorted Generalization (Extended SG) Algorithm

Input: n sorted atoms, $\alpha_i = A_i/a_{i,1} \wedge \dots \wedge a_{i,m_i}$ for $1 \leq i \leq n$, and a sort theory Σ .
Output: An msg_Σ of $\alpha_1, \dots, \alpha_n$.

1. Let Head = Unsorted-Generalization(A_1, \dots, A_n).
2. If Head = *nil*, then return *nil*.
3. Initialize Constraint-Set to the empty set.
4. For each term u in Head:
 - a. Let t_i be the corresponding term in A_i , for all $1 \leq i \leq n$.
 - b. Let Min-Sorts be the minimal (under the \preceq_Σ ordering) members of $\{\tau \mid \Sigma \models \bar{V}(a_{i,1} \wedge \dots \wedge a_{i,m_i} \rightarrow \tau(t_i)) \text{ for all } 1 \leq i \leq n\}$.
 - c. For each sort τ in Min-Sorts, add the constraint formula $\tau(u)$ to Constraint-Set.
5. Return the sorted atom whose head is Head and whose constraint is the conjunction of the constraint formulas in Constraint-Set.

Figure 1: The Extended Sorted Generalization (SG) Algorithm

a finite set of extended sorted atoms whose total sizes sum to S and which contains at least one atom of head size no greater than l in time $O(lQ(S))$.

Proof: [Time Complexity] The time for the ordinary generalization step is $O(H)$ where H is the sum of the head sizes of the sorted atoms being generalized. The time for computing term correspondences (for example, establishing pointers from each term in the ordinary generalization, the msg of the heads, to the corresponding terms) is also $O(H)$. The size of the ordinary generalization can be no greater than the least head size among the sorted atoms being generalized. Therefore, there are at most l terms in the ordinary generalization. For each of these l terms, the algorithm uses the corresponding terms to build a taxonomic query. Because term correspondences have already been computed, any taxonomic query can be built in time linear in the size of the query, which can be at most S . The time for a query to be made and answered must be at least linear in the size of the query and the number of sorts in Σ (the size of the largest answer), so the term $Q(S)$ dominates the time spent in building the query. Because $S \geq H$, this term also dominates the time spent in ordinary generalization and in computing corresponding terms. \square

3.2 Constraint Generalization

Many interesting concepts require constraints *among* terms. Our present language can only represent monadic constraints, or sorts, *on* single terms. Thus sorted atoms as presently defined cannot represent, for example, the concept that electric current flows from any one object x to any other object y given the constraints that x is metallically connected to y and that x has a higher electrical potential than does y .

We overcome this shortcoming as follows. In place of a set of distinguished monadic predicates, which we have

been calling sorts, we use a set of distinguished predicates of arbitrary arity. We call these predicates *constraint predicates*. Analogous to sort theories, a constraint theory is any theory whose only predicates are constraint predicates. Rather than sorted atoms, we now have constrained atoms, or atoms with constraints *among* terms in addition to monadic constraints, or sorts, *on* terms. The definition of \geq_Σ remains unchanged.

Following are several examples of the \geq_Σ relation for constrained atoms. Some of these examples use the following constraint theory, Σ_2 .

$$\begin{aligned} \Sigma_2 = \{ & \text{BIGGER}(\text{son}(\text{jumbo}), \text{son}(\text{clyde})), \\ & \text{BIGGER}(\text{son}(\text{fred}), \text{son}(\text{joe})), \\ & \text{BIGGER}(\text{jumbo}, \text{clyde}), \\ & \text{BIGGER}(\text{fred}, \text{joe}) \}, \\ & \text{ELEPHANT}(\text{clyde}), \text{ELEPHANT}(\text{jumbo}), \\ & \text{HUMAN}(\text{fred}), \text{HUMAN}(\text{joe}), \\ & \forall x \forall y \text{ ELEPHANT}(x) \wedge \text{HUMAN}(y) \rightarrow \\ & \text{BIGGER}(x, y) \}. \end{aligned}$$

Example 14

$$\begin{aligned} & \text{intimidates}(\text{son}(x_1), \text{son}(y_1)) / \\ & \quad \text{BIGGER}(\text{son}(x_1), \text{son}(y_1)) \\ \geq_{\Sigma_2} & \text{intimidates}(\text{son}(x_2), \text{son}(y_2)) / \\ & \quad \text{BIGGER}(x_2, y_2) \wedge \text{BIGGER}(\text{son}(x_2), \text{son}(y_2)) \\ \geq_{\Sigma_2} & \text{intimidates}(\text{son}(\text{jumbo}), \text{son}(\text{clyde})). \end{aligned}$$

To verify that the first relation holds, first note that any substitution θ mapping x_1 to x_2 and y_1 to y_2 also maps $\text{intimidates}(\text{son}(x_1), \text{son}(y_1))$ to $\text{intimidates}(\text{son}(x_2), \text{son}(y_2))$. We then need to see if Σ_2 entails

$$\begin{aligned} & \bar{V}(\text{BIGGER}(x_2, y_2) \wedge \text{BIGGER}(\text{son}(x_2), \text{son}(y_2)) \\ & \quad \rightarrow (\text{BIGGER}(\text{son}(x_1), \text{son}(y_1)))\theta). \end{aligned}$$

Constraint Generalization Algorithm

Input: n constrained atoms, $\alpha_i = A_i/a_{i,1} \wedge \dots \wedge a_{i,m_i}$ for $1 \leq i \leq n$, and a constraint theory Σ .
Output: An msg_Σ of $\alpha_1, \dots, \alpha_n$.

1. Let Head = Ordinary-Generalization(A_1, \dots, A_n).
2. If Head = *nil*, then return *nil*.
3. Initialize Constraint-Set to the empty set.
4. Let r be the maximum of the arities of all constraint predicate in Σ .
 For all $1 \leq l \leq r$ and each tuple (each permutation of each set) $\langle u_1, \dots, u_l \rangle$ of l terms in Head:
 - a. Let $\langle t_{i,1}, \dots, t_{i,l} \rangle$ be the tuple of corresponding terms in A_i , for all $1 \leq i \leq n$.
 - b. Let Min-Constraint-Predicates be the minimal (under the \preceq_Σ ordering) members of $\{\tau \mid \tau \text{ has arity } l, \text{ and } \Sigma \models \bar{\forall}(a_{i,1} \wedge \dots \wedge a_{i,m_i} \rightarrow \tau(t_{i,1}, \dots, t_{i,l})) \text{ for all } 1 \leq i \leq n\}$.
 - c. For each constraint predicate τ in Min-Constraint-Predicates, add the constraint formula $\tau(u_1, \dots, u_l)$ to Constraint-Set.
5. Return the constrained atom whose head is Head and whose constraint is the conjunction of the constraint formulas in Constraint-Set.

Figure 2: The Constraint Generalization Algorithm

It does, because

$$\begin{aligned} & \text{BIGGER}(\text{son}(x_1), \text{son}(y_1))\theta \\ &= (\text{BIGGER}(\text{son}(x_2), \text{son}(y_2)), \end{aligned}$$

and Σ_2 entails

$$\begin{aligned} & \bar{\forall}(\text{BIGGER}(x_2, y_2) \wedge \text{BIGGER}(\text{son}(x_2), \text{son}(y_2))) \\ & \rightarrow \text{BIGGER}(\text{son}(x_2), \text{son}(y_2)). \end{aligned}$$

To see that the second relation holds, consider a substitution θ that maps x_2 to jumbo and y_2 to clyde. First, notice that

$$\begin{aligned} & \text{intimidates}(\text{son}(x_2), \text{son}(y_2))\theta = \\ & \text{intimidates}(\text{son(jumbo}), \text{son(clyde)}). \end{aligned}$$

Second, note that

$$\begin{aligned} & (\text{BIGGER}(x_2, y_2) \wedge \\ & \quad \text{BIGGER}(\text{son}(x_2), \text{son}(y_2)))\theta \\ &= \text{BIGGER}(\text{son(jumbo}), \text{son(clyde)}) \wedge \\ & \quad \text{BIGGER}(\text{son(jumbo)}, \text{son(clyde)}), \end{aligned}$$

and that Σ_2 entails

$$\begin{aligned} & \text{BIGGER}(\text{son(jumbo}), \text{son(clyde)}) \wedge \\ & \text{BIGGER}(\text{son(jumbo)}, \text{son(clyde)}). \end{aligned}$$

The only change required to the Extended SG Algorithm to deal with the more general language is in step 4, that of computing the constraints. Rather than compute the sorts to place on each term in the head of the generalization, the algorithm must instead determine which constraints to place on each tuple of terms in the head. The algorithm that results from this modification is the Constraint Generalization Algorithm (Figure 2). Note that the algorithm need not

consider tuples of more than r terms, where r is the maximum of the arities of all constraint predicates in Σ .

For the presentation of the algorithm, it is useful to extend the \preceq_Σ relation defined on sorts to apply to arbitrary constraint predicates of the same arity. For n -ary constraint predicates τ_1 and τ_2 , we say that $\tau_1 \preceq_\Sigma \tau_2$ if and only if $\Sigma \models \bar{\forall}(\tau_1(x_1, \dots, x_n) \rightarrow \tau_2(x_1, \dots, x_n))$.

The Constraint Generalization Algorithm uses constraint queries, rather than taxonomic queries, to interact with the constraint theory. The algorithm needs to know the maximum of the arities of all constraint predicates in Σ . It also needs the following oracle, which is stronger than that used by the Extended SG Algorithm. For all $1 \leq i \leq n$, let $a_{i,1} \wedge \dots \wedge a_{i,m_i}$ be a constraint and let $\langle t_{i,1}, \dots, t_{i,j} \rangle$ be a tuple of j terms, where j is less than the maximum of the arities of all constraint predicates in Σ . The oracle returns the members of the set $\{\tau \mid \tau \text{ has arity } l, \text{ and } \Sigma \models \bar{\forall}(a_{i,1} \wedge \dots \wedge a_{i,m_i} \rightarrow \tau(t_{i,1}, \dots, t_{i,l})) \text{ for all } 1 \leq i \leq n\}$ that are minimal under the \preceq_Σ ordering. This oracle is in fact necessary for computation of constraint generalization by any algorithm if we want to exclude redundancy *among constraint predicates of the same arity*. Again, if we do not wish to exclude such redundancy, an obvious, weaker oracle is necessary and sufficient. Because the algorithm does not compare constraint predicates of different arity, the constraint on the constrained atom it returns may contain some redundancy, but the constrained atom is still an msg_Σ . The following examples shows msg_Σ s the algorithm computes.

Example 15

Let E be

$\{\text{intimidates}(\text{son}(\text{jumbo}), \text{son}(\text{clyde})),$
 $\quad \text{intimidates}(\text{son}(\text{fred}), \text{son}(\text{joe}))\}$.

An msg_{Σ_2} of E is:

$\text{intimidates}(\text{son}(x_2), \text{son}(y_2)) /$
 $\quad \text{BIGGER}(x_2, y_2) \wedge$
 $\quad \text{BIGGER}(\text{son}(x_2), \text{son}(y_2))$.

Example 16

Let E be

$\{\text{intimidates}(\text{clyde}, y_1) / \text{HUMAN}(y_1),$
 $\quad \text{intimidates}(x_1, \text{fred}) / \text{ELEPHANT}(x_1)\}$.

An msg_{Σ_2} of E is:

$\text{intimidates}(x_2, y_2) / \text{BIGGER}(x_2, y_2) \wedge$
 $\quad \text{ELEPHANT}(x_2) \wedge \text{HUMAN}(y_2)$.

Theorem 17 Let Σ be a constraint theory such that the maximum of the arities of all constraint predicates in Σ is r , and any constraint query about Σ having size s is made and answered in time $Q(s)$. The Constraint Generalization Algorithm computes an msg_{Σ} of n constrained atoms whose total sizes sum to S and which contains at least one atom with head size no greater than l in time $O(r!^r Q(rS))$.

Proof:[Time Complexity] The analysis is the same as that for Theorem 13 except for two points. First, the size of a constraint query can be almost as large as rS , rather than S , because the query uses tuples of up to r terms, some of which may appear as subterms of others.⁷ Second, rather than querying the oracle for each term in the ordinary generalization, the algorithm must query the oracle for each tuple of 1 to r terms, the number of which is bounded by $r!^r$. \square

4 Applications

Applications of sorted generalization to abduction, induction, analogical reasoning, and knowledge base vivification are based on semantic properties of the \geq_{Σ} relation (Section 2 and [Frisch and Page, 1990]). These same properties hold for the extended \geq_{Σ} relation for extended sorted generalization and constraint generalization.

Theorem 18 Let Σ be a constraint theory and let α_1 and α_2 be constrained atoms. $\alpha_1 \geq_{\Sigma} \alpha_2$ if and only if $\{\exists \alpha_1\} \cup \Sigma \models \exists \alpha_2$.

⁷Suppose, for example, that r is 2, and the constrained atoms being generalized have empty constraints (so $H = S$). Suppose also that the heads of the atoms each have one argument (of size S/n), each of which consists of a function applied to an argument of size $(S/n) - 1$. Then there is a query of size $2S - n$. So if three constrained atoms are being generalized, the query has size $rS - 3$, where $r = 2$.

Theorem 19 Let Σ be a constraint theory and let α_1 and α_2 be constrained atoms. $\alpha_1 \geq_{\Sigma} \alpha_2$ if and only if $\{\exists \alpha_2\} \cup \Sigma \models \exists \alpha_1$.

Theorem 18 justifies the use of sorted generalization for abduction and induction. Theorem 19 makes sorted generalization applicable to knowledge base vivification. Sorted generalization is applicable to analogical reasoning, based on both theorems. Rather than repeat arguments made elsewhere [Frisch and Page, 1990], we further investigate the application to induction. More specifically, we ask whether the Constraint Generalization Algorithm can be used for polynomial learning by equivalence queries and polynomial pac-learning. The definitions of these learning models for constrained atoms are the same as for sorted atoms (Section 2), except Σ may be any constraint theory, rather than sort theory, and queries are constraint queries, rather than taxonomic queries.

If we add to the set of constrained atoms a weakest element that (with any constraint theory Σ) covers no constrained atoms, then the following algorithm polynomially learns constrained atoms by equivalence queries with respect to any fixed constraint theory Σ . The algorithm first predicts the weakest element. If it is correct, the algorithm is finished. If it is incorrect, the only counterexamples are constrained atoms labeled as positive examples. One positive example is received in response to the prediction. From this point on, the algorithm predicts the msg_{Σ} of the positive examples received thus far. Because the msg_{Σ} is the weakest constrained atom Σ -more general than the positive examples seen thus far, it is always weaker than or equivalent to the target constrained atom. Therefore the algorithm receives only positive examples as counterexamples.

Because the Constraint Generalization Algorithm runs in time polynomial in the sizes of the constrained atoms being generalized, each prediction can be computed in time polynomial in the sizes of the counterexamples seen thus far. The following theorem (the proof of which is in the appendix) shows that for fixed Σ the algorithm makes at most a number of mistakes polynomial in the size of the first counterexample the algorithm receives. After this many mistakes, the algorithm predicts correctly. This result completes the argument that the algorithm polynomially learns constrained atoms by equivalence queries for any fixed Σ , since the total time for the algorithm to reach the correct prediction is the product of the number of mistakes and the maximum time to compute a prediction.

Theorem 20 Let C be the number of constraint predicates in Σ , and let r be the maximum of the arities of all constraint predicates in Σ . The maximum chain $\alpha_1 \geq_{\Sigma} \dots \geq_{\Sigma} \alpha_m$ of constrained atoms of head size at most n has length (m) at most $Cn^r + n + 1$.

Because each prediction of the learning algorithm is strictly Σ -more general than the previous prediction, and the maximum length of such a chain of predictions is $Cn^r + n + 1$, the algorithm makes at most $Cn^r + n + 1$ prediction mistakes (counting the original prediction of the element that covers no constrained atoms). C and r are constant for a fixed constraint theory, so the number of prediction mistakes is polynomial in n .

As Section 2 notes, polynomial learning by equivalence queries implies polynomial pac-learning. More specifically, any algorithm that computes a constrained atom consistent with a set of constrained atoms, where each atom is labeled as a positive or negative example, in time polynomial in the sizes of the examples can be used to pac-learn constrained atoms. The Constraint Generalization Algorithm can be used for such computations with respect to any fixed constraint theory, Σ , for which we have polynomial-time response to constraint queries. To do so, first use the Constraint Generalization Algorithm to compute the msg_Σ of the positive examples. If any constrained atom is consistent with the set of examples, the msg_Σ of the positive examples is consistent. Then check whether the msg_Σ covers any negative examples. This check can be done in polynomial time using the same constraint queries used by the Constraint Generalization Algorithm. If the msg_Σ covers no negative examples, then it is consistent with the set of examples. Otherwise, no constrained atom is consistent with the set of examples; that is, the target concept is not a constrained atom.

The mistake bound for polynomial learning by equivalence queries gives us a sufficient number of examples to use for a given desired accuracy and confidence in pac-learning [Angluin, 1988b]. The next result, which is also achieved using the size of a maximum chain of constrained atoms, provides a tighter bound on the number of examples needed for pac-learning. To present this result, we need the following definition that applies the notion of *VC-dimension* [Blumer *et al.*, 1989] to constrained atoms.

Definition 21 *The VC-dimension of constrained atoms of head size at most n with respect to a constraint theory, Σ , is the size of the largest set S of such atoms for which the following holds. For any subset s of S , there exists a constrained atom that, with Σ , covers every constrained atom in s and covers no constrained atoms in $S - s$. Such a set S is said to be shattered.*

Theorem 22 *Let Σ be a constraint theory having C constraint predicates, and let the maximum of the arities of all constraint predicates in Σ be r . The VC-dimension, with respect to Σ , of constrained atoms having head size at most n is $Cn^r + n + 1$.*

A sketch of the proof of this result is in the appendix. Based on work in Computational Learning

Theory[Blumer *et al.*, 1989], this result on the VC-dimension of constrained atoms implies that the number of examples needed for pac-learning is

$$\max\left(\frac{4}{\epsilon} \log \frac{2}{\delta}, \frac{8d}{\epsilon} \log \frac{13}{\epsilon}\right)$$

where d is $Cn^r + n + 1$ and n is the head size of the target concept. Because C and r are constants for a fixed constraint theory, Σ , the VC-dimension is polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and n .⁸

In general, the learnability of a class of concepts, such as constrained atoms, does not imply the learnability of a subclass, such as ordinary atoms or extended sorted atoms. In this case, however, the proof of the VC-dimension of constrained atoms applies to ordinary and extended sorted atoms as well. Therefore, the VC-dimension of ordinary atoms, for which the constraint theory is always empty, is $n+1$, and the VC-dimension of extended sorted atoms, for which the constraint theory can only contain monadic constraints, is $Cn+n+1$. The Ordinary Generalization and Extended SG Algorithms can be used to polynomially pac-learn ordinary atoms and extended sorted atoms (with respect to a fixed sort theory), respectively, in the same way the Constraint Generalization Algorithm is used to polynomially pac-learn constrained atoms.

5 Relationship to Other Work

Throughout most of the paper we have viewed sorted and constraint generalization as operations on atoms in constraint logic. We have noted that sorted and constrained atoms, universally closed, are equivalent to special kinds of definite clauses. This equivalence lets us relate these operations to the *generalized subsumption* of Buntine [1988]. Generalized subsumption is defined over definite clauses with respect to a logic program that acts as the background information. Because constrained atoms are equivalent to definite clauses whose antecedents, or bodies, contain only constraints on terms in the consequent, or head, we can compare Σ -subsumption, on which sorted and constraint generalization are based, with generalized subsumption, in cases where Σ consists entirely of definite clauses. In these cases, Σ -subsumption and generalized subsumption are equivalent. In the case where Σ -subsumption is based on information that comprises a constraint theory for which constraint queries are answered efficiently, the Constraint Generalization Algorithm can be used to efficiently generalize constrained atoms. Thus our work may be seen as identifying a restricted case for which Σ -subsumption is efficiently computable. An interesting area for further work is the search for other ways to extend the languages of

⁸See [Anthony *et al.*, 1990] for another VC-dimension-based bound upper bound on the number of examples needed for pac-learning.

atoms and background information while constraining it enough to preserve polynomial-time generalization based on Σ -subsumption.

Appendix

5.1 Proof of Theorem 20

First, we extend the definition of \preceq_Σ . As presently defined, $\tau_1 \preceq_\Sigma \tau_2$ if and only if constraint predicates τ_1 and τ_2 have the same arity, n , and $\Sigma \models \bar{\forall}(\tau_1(x_1, \dots, x_n) \rightarrow \tau_2(x_1, \dots, x_n))$. We now define \preceq_Σ between sets of constraint predicates of the same arity in an analogous fashion: $\{\tau_1, \dots, \tau_m\} \preceq_\Sigma \{\omega_1, \dots, \omega_k\}$ if and only if $\tau_1, \dots, \tau_m, \omega_1, \dots, \omega_k$ all have the same arity, n , and $\Sigma \models \bar{\forall}(\tau_1(x_1, \dots, x_n) \wedge \dots \wedge \tau_m(x_1, \dots, x_n) \rightarrow \omega_1(x_1, \dots, x_n) \wedge \dots \wedge \omega_k(x_1, \dots, x_n))$. We also say in this case that the first set of constraints is *tighter than* the second. Note that for a constraint theory, Σ , containing C constraint predicates, the maximum chain of sets of constraint predicates ordered by \preceq_Σ has length at most C . This is because any set in such a chain must contain some predicate that does not appear in a tighter set (a set below it) in the chain.

We now show that there can be no chain of length $Cn^r + n + 2$. We associate a *height* with each constrained atom α . Let m be the number of variables (not variable occurrences) in the head of α , and let k be the number of variable and function symbol occurrences in the head of α . For each tuple $T = \{t_1, \dots, t_j\}$ of $j \leq r$ terms in the head of α , if T is unconstrained then let $\#(T) = C$. Otherwise let N be the set of constraint predicates applied to T in the constraint of α ; let $\#(T)$ be the maximum number of sets of constraint predicates that are below N in a chain of such sets. Let $\text{height}(\alpha)$ be

$$(n - k) + m + 1 + \sum_{\text{tuples } T \text{ of } j \leq r \text{ terms}} \#(T).$$

We first show that the size of a maximum chain of constrained atoms, each with head size no larger than n , is the maximum height of any constrained atom of head size at most n . This is shown by proving that the size of a maximum chain having greatest element α is at most $\text{height}(\alpha)$. The proof is by induction on $\text{height}(\alpha)$. An observation that is useful in the proof is that α is Σ -more general than another constrained atom, β , if and only if β results from tightening the set of constraint predicates on one or more tuples of terms in α or from substituting non-variable terms for one or more variables in α . This observation is straightforward given the definition of Σ -more general.

Basis: Height is 1. Then no sets of constraint predicates on tuples of terms in α can be tightened, and α contains no variables, so no substitutions can be ap-

plied. Therefore, α is at the bottom of every chain in which it appears.

Inductive: Tightening the set of constraint predicates on a tuple T of terms in α reduces $\#(T)$. Tightening a set of constraint predicates therefore reduces $\text{height}(\alpha)$. It remains to show that substituting a non-variable term for a variable, x , reduces the height. Such a substitution can only change m or k . Note that any non-variable term of length i can contain at most $i - 1$ variables. Substituting such a term for x removes x from α and adds to α at most $i - 1$ variables. Therefore, applying such a substitution increases m by at most $i - 2$ and decreases k by $i - 1$. Hence applying such a substitution reduces $\text{height}(\alpha)$.

To complete the proof, we show that the maximum value for $\text{height}(\alpha)$ is $Cn^r + n + 1$. Because there are at most n r -tuples of terms in α , and the length of the largest chain of constraints on each of these is C , the maximum value of

$$\sum_{\text{tuples } T \text{ of } j \leq r \text{ terms}} \#(T)$$

is Cn^r . Because m is at most k , the maximum value of $(n - k) + m + 1$ is $n + 1$. Therefore, the maximum value for $\text{height}(\alpha)$ is $Cn^r + n + 1$. \square

[Notice that the size of a maximum chain does not increase if we allow the head, B , of the bottom constrained atom, β , to have head size greater than n but still require all others to have head size at most n . To see this, note that if B has head size greater than n , then it has head size greater than the size of the head (B') of the atom, β' , above it. Then B' is not the same as B , but for some substitution θ , $B'\theta = B$. Then B' must have at least one variable, so the height of β' must be at least 2. But the height of the top element, α , in the chain can be at most $Cn^r + n + 1$, so the chain from α to β' , inclusive, has length at most $Cn^r + n$. This result is useful in the proof of Theorem 22.]

We note that for at least some constraint theories, chains of length $Cn^r + n + 1$ do exist. Let Σ be a constraint theory whose constraint predicates are τ_1, \dots, τ_C , each of arity r , such that $\tau_C \prec_\Sigma \dots \prec_\Sigma \tau_1$. Let the greatest element in the chain be $p(x_1, \dots, x_n)$. To get the each of the next n^r atoms of the chain, add to the constraint of the previous atom an atomic formula built from the predicate τ_1 and one of the n^r r -tuples of variables from x_1, \dots, x_n not used with τ_1 thus far. Repeat for each constraint τ_2, \dots, τ_C to build a chain of total length $Cn^r + 1$. Let the remaining n elements of the chain be the result of replacing, one at a time, the variables in the head with constants.

5.2 Proof Sketch for Theorem 22

We begin by showing that no set of $Cn^r + n + 2$ constrained atoms is shattered by constrained atoms of

head size at most n . We do so by showing that if such a set were shattered, there would exist a chain of constrained atoms, all but the bottom of which have head size at most n , of length $Cn^r + n + 2$.⁹ This would directly contradict the generalization of Theorem 20 given after the preceding proof. Let $\{\alpha_1, \dots, \alpha_{Cn^r+n+2}\}$ be the set. Because the set is shattered, we can note that for all i between 1 and $Cn^r + n + 2$, inclusive, a constrained atom exists that covers $\alpha_1, \dots, \alpha_i$ and covers no other members of the set. Then by the definition of msg_Σ , the msg_Σ, g_i , of $\alpha_1, \dots, \alpha_i$ is such a constrained atom. Then $g_{Cn^r+n+2} >_\Sigma \dots >_\Sigma g_1$, yielding a chain of size $Cn^r + n + 2$.

We now give a simple example showing that for some constraint theories, a set of $Cn^r + n + 1$ constrained atoms can be shattered by constrained atoms whose heads have size at most n . Let Σ be the empty constraint theory. Then C is 0, so $Cn^r + n + 1 = n + 1$. It is enough to present a shattered set of size $n + 1$ for a particular n . We present the set for $n = 4$, from which it is easy to also identify such sets for all other values of n . The set $\{p(a, a, a, a), p(b, a, a, a), p(b, b, a, a), p(b, b, b, a), p(b, b, b, b)\}$ is shattered by constrained atoms of size at most 4. For each atom in the set, there is a way to exclude, or fail to cover, only that atom, and these ways of exclusion are orthogonal. For example, any atom that has a b for the first argument fails to cover the first atom. Any atom whose first two arguments are the same fails to cover the second atom. Any atom whose second and third arguments are the same fails to cover the third atom. Because these are orthogonal, they can be combined to exclude any subset of the atoms while including all others. Here are two examples of such combinations. The atom $p(x, y, y, y)$ excludes the third and fourth atoms because the second, third, and fourth arguments are the same. The atom $p(b, b, x, x)$ excludes the first atom because the first argument is a b , the second atom because the first and second arguments are the same, and the fourth atom because the third and fourth arguments are the same. For any subset we construct an atom that covers the atoms in the subset and excludes the others. Thus the set of five atoms is shattered by atoms of (head) size 4. \square

References

- [Angluin, 1988a] D. Angluin. *Negative Results for Equivalence Queries*. Technical Report YALEU/DCS/RR-648, Yale University, September 1988.
- [Angluin, 1988b] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.
- [Anthony et al., 1990] Martin Anthony, Norman Biggs, and John Shawe-Taylor. The learnability of formal concepts. In *Proceedings of the 1990 Workshop on Computational Learning Theory*, pages 246–257, Morgan Kaufmann, Rochester, NY, August 1990.
- [Blumer et al., 1989] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *JACM*, 36(4):929–965, 1989.
- [Buntine, 1988] W. Buntine. Generalized subsumption and its applications to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.
- [Frisch and Page, 1990] A. M. Frisch and C. D. Page. Generalization with taxonomic information. In *Proceedings of AAAI-90*, pages 755–761, Boston, MA, 1990.
- [Lassez and Marriott, 1987] J-L. Lassez and K. Marriott. Explicit representation of terms defined by counter examples. *Journal of Automated Reasoning*, 3:301–317, 1987.
- [Lassez et al., 1988] J-L. Lassez, M. J. Maher, and K. Marriott. Unification revisited. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, chapter 15, pages 587–625, Morgan Kaufmann Publishers, 1988.
- [Mitchell et al., 1983] T. G. Mitchell, P. E. Utgoff, and R. Banerji. Learning by experimentation: Acquiring and redefining problem solving heuristics. In T.M. Mitchell R.S. Michalski, J.G. Carbonell, editor, *Machine Learning: An Artificial Intelligence Approach*, chapter 5, pages 137–162, Morgan Kaufmann Publishers, 1983.
- [Pitt and Valiant, 1988] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *JACM*, 35(4):965–984, 1988.
- [Plotkin, 1970] G. D. Plotkin. *A Note on Inductive Generalization*, chapter 8, pages 153–163. Volume 5 of *Machine Intelligence*, Edinburgh University Press, Edinburgh, 1970.
- [Plotkin, 1971] G. D. Plotkin. *A Further Note on Inductive Generalization*, chapter 8, pages 101–124. Volume 6 of *Machine Intelligence*, Edinburgh University Press, 1971.
- [Reynolds, 1970] J. C. Reynolds. *Transformational Systems and Algebraic Structure of Atomic Formulas*, chapter 7, pages 135–52. Volume 5 of *Machine Intelligence*, Edinburgh University Press, Edinburgh, 1970.
- [Valiant, 1984] L. G. Valiant. A theory of the learnable. *Comm. ACM*, 27(11):1134–1142, 1984.

⁹We thank Professor Douglas West of the University of Illinois Mathematics Department for a helpful conversation, that led us to consider this method of proof about VC-dimension within partially ordered sets.

A Theory of Inferred Causation

Judea Pearl
< judea@cs.ucla.edu >
 Cognitive Systems Laboratory
 Computer Science Department
 University of California
 Los Angeles, CA 90024

T.S. Verma
< verma@cs.ucla.edu >
 Cognitive Systems Laboratory
 Computer Science Department
 University of California
 Los Angeles, CA 90024

Abstract

This paper concerns the empirical basis of causation, and addresses the following issues:

1. the clues that might prompt people to perceive causal relationships in uncontrolled observations.
2. the task of inferring causal models from these clues, and
3. whether the models inferred tell us anything useful about the causal mechanisms that underly the observations.

We propose a minimal-model semantics of causation, and show that, contrary to common folklore, genuine causal influences can be distinguished from spurious covariations following standard norms of inductive reasoning. We also establish a sound characterization of the conditions under which such a distinction is possible. We provide an effective algorithm for inferred causation and show that, for a large class of data the algorithm can uncover the direction of causal influences as defined above. Finally, we address the issue of non-temporal causation.

1 Introduction

The study of causation is central to the understanding of human reasoning. Tasks involving changing environments require causal theories which make formal distinctions between causation and logical implication [Geffner, 1989, Lifschitz, 1987, Pearl, 1988a, Shoham, 1988]. In applications such as diagnosis [Patil et al., 1982, Reiter, 1987], qualitative physics [Bobrow, 1985], and plan recognition [Kautz, 1987, Wilensky, 1983], a central task is that of finding a satisfactory *explanation* to a given set of observations, and the meaning of explanation is intimately related to the notion of causation.

Most AI works have given the term "cause" a procedural semantics, attempting to match the way people use it in reasoning tasks, but were not concerned with the experience that prompts people to believe that "*a causes b*", as opposed to, say, "*b causes a*" or "*c causes both a and b*". The question of choosing an appropriate causal ordering received some attention in qualitative physics, where certain interactions attain directionality despite the instantaneous and symmetrical nature of the underlying equations, as in "current causing a voltage drop across the resistor" [Forbus and Gentner, 1986]. In some systems causal ordering is defined as the ordering at which subsets of variables can be solved independently of others [Iwasaki and Simon, 1986], in other systems it follows the way a disturbance is propagated from one variable to others [de Kleer and Brown, 1986]. Yet these choices are made as a matter of convenience, to fit the structure of a given theory, and do not reflect features of the empirical environment which compelled the formation of the theory.

An empirical semantics for causation is important for several reasons. First, an intelligent system attempting to build a workable model of its environment cannot rely exclusively on preprogrammed causal knowledge, but must be able to translate direct observations to cause-and-effect relationships. Second, by tracing empirical origins we stand to obtain an independent gauge for deciding which of the many logics proposed for causal reasoning is sound and/or complete, and which provides a proper account of causal utterances such as "*a explains b*", "*a suggests b*", "*a tends to cause b*", and "*a actually caused b*", etc.

While the notion of causation is often associated with those of necessity and functional dependence, causal expressions often tolerate exceptions, primarily due to missing variables and coarse description. We say, for example, "reckless driving causes accidents" or "you will fail this course because of your laziness". Suppes [Suppes, 1970] has argued convincingly that most causal utterances in ordinary conversation reflect prob-

abilistic, not categorical relations¹. Thus, probability theory should provide a natural language for capturing causation [Reichenbach, 1956, Good, 1983]. This is especially true when we attempt to infer causation from (noisy) observations – probability calculus remains an unchallenged formalism when it comes to translating statistical data into a system of revisable beliefs.

However, given that statistical analysis is driven by covariation, not causation, and assuming that most human knowledge derives from statistical observations, we must still identify the clues that prompt people to perceive causal relationships in the data, and we must find a computational model that emulates this perception.

Temporal precedence is normally assumed essential for defining causation, and it is undoubtedly one of the most important clues that people use to distinguish causal from other types of associations. Accordingly, most theories of causation invoke an explicit requirement that a cause precedes its effect in time [Good, 1983, Reichenbach, 1956, Shoham, 1988, Suppes, 1970]. Yet temporal information alone cannot distinguish genuine causation from spurious associations caused by unknown factors. In fact the statistical and philosophical literature has adamantly warned analysts that, unless one knows in advance all causally relevant factors, or unless one can carefully manipulate some variables, no genuine causal inferences are possible [Cartwright, 1989, Cliff, 1983, Eells and Sober, 1983, Fisher, 1953, Gardenfors, 1988, Holland, 1986, Skyrms, 1986]². Neither condition is realizable in normal learning environments, and the question remains how causal knowledge is ever acquired from experience.

This paper introduces a minimal-model semantics of causation which provides a plausible account for how causal models could be inferred from observations. Using this semantics we show that genuine causal influences can in many cases be distinguished from spurious covariations and, moreover, the direction of causal influences can often be determined without resorting to chronological information. (Although, when available, chronological information can significantly simplify the modeling task.) Such semantics should be applicable, therefore, to the organization of concurrent events or events whose chronological precedence cannot be determined with precision, (e.g. “old age explains disabilities”) in the spirit of Glymour [Glymour et al., 1987] and Simon [Simon, 1954].

¹See [Dechter and Pearl, 1990] for a treatment of causation in the context of categorical data.

²Some of the popular quotes are: “No causation without manipulation”, [Holland, 1986], “No causes in, no causes out”, [Cartwright, 1989] “No computer program can take account of variables that are not in the analysis”, [Cliff, 1983].

This paper is organized as follows. In Section 2 we define the notions of causal models and causal theories, and describe the task of causal modeling as an identification game scientists play against Nature. In Section 3 we introduce the minimal-model semantics of causation and exemplify its operability and plausibility on a simple example. Section 4 identifies conditions under which effective algorithms exist that uncover the structure of causal influences as defined above. One such algorithm (called IC) is introduced in Section 5, and is shown to be sound for the class of stable distributions, even when some variables are not observable³. Section 6 extracts from the IC-algorithm the essential conditions under which causal influences are identified and proposes these as independent definitions of genuine influences and spurious associations, with and without temporal information. Section 7 provides an intuitive justification for the definitions proposed in Section 6, showing that our theory conforms to the common understanding of causation as a stipulation of stable behavior under external interventions. The definitions are shown to be in line with accepted standards of controlled experimentation, save for requiring the identification of “virtual” experimental conditions within the data itself. In Section 8 we invoke the “virtual control” metaphor to elucidate how causal relationships can still be ascertained in the absence of temporal information. We then offer an explanation for the puzzling, yet universal agreement between the temporal and the statistical aspects of causation.

2 The Causal Modeling Framework

We view the task of causal modeling as an identification game which scientists play against Nature. Nature possesses stable causal mechanisms which, on a microscopic level are deterministic functional relationships between variables, some of which are unobservable. These mechanisms are organized in the form of an acyclic schema which the scientist attempts to identify.

Definition 1 A causal model of a set of variables U is a directed acyclic graph (dag), in which each node corresponds to a distinct element of U .

The nodes of the dag correspond to the variables under analysis, while the links denote direct causal influences among the variables. The causal model serves as a blueprint for forming a “causal theory” – a precise specification of how each variable is influenced by its parents in the dag. Here we assume that Nature is at liberty to impose arbitrary functional relationships between each effect and its causes and then to perturb these relationships by introducing arbitrary (yet mutually independent) disturbances. These disturbances reflect “hidden” or unmeasurable conditions and exceptions

³Proofs can be found in [Verma, 1991].

which Nature chooses to govern by some undisclosed probability function.

Definition 2 A causal theory is a pair $T = \langle D, \Theta_D \rangle$ consisting of a causal model D and a set of parameters Θ_D compatible with D . Θ_D assigns a function $x_i = f_i[\text{pa}(x_i), \epsilon_i]$ and a probability measure g_i , to each $x_i \in U$, where $\text{pa}(x_i)$ are the parents of x_i in D and each ϵ_i is a random disturbance distributed according to g_i , independently of the other ϵ 's and of any preceding variable $x_j : 0 < j < i$

This requirement of independence renders the disturbances "local" to each parents-child family; disturbances that influence several families simultaneously will be treated explicitly as "latent" variables (see Definition 3).

Once a causal theory T is formed, it defines a joint probability distribution $P(T)$ over the variables in the system, and this distribution reflects some features of the causal model (e.g., each variable must be independent of its grandparents, given the values of its parents). Nature then permits the scientist to inspect a select subset $O \subseteq U$ of "observed" variables, and to ask questions about the probability distribution over the observables, but hides the underlying causal theory as well as the structure of the causal model. We investigate the feasibility of recovering the topology of the dag, D , from features of the probability distribution.⁴

3 Model preferences (Occam's razor)

In principle, U being unknown, there is an unbounded number of models that would fit a given distribution, each invoking a different set of "hidden" variables and each connecting the observed variables through different causal relationships. Therefore with no restriction on the type of models considered, the scientist is unable to make any meaningful assertions about the structure underlying the phenomena. Likewise, assuming $U = O$ but lacking temporal information, he/she can never rule out the possibility that the underlying model is a complete (acyclic) graph; a structure that, with the right choice of parameters can mimic (see Definition 4) the behavior of any other

⁴This formulation invokes several idealizations of the actual task of scientific discovery. It assumes, for example, that the scientist obtains the distribution directly, rather than events sampled from the distribution. This assumption is justified when a large sample is available, sufficient to reveal all the dependencies embedded in the distribution. Additionally, we assume that the observed variables actually appear in the original causal theory and are not some aggregate thereof. Aggregation might result in feedback loops which we do not discuss in this paper. Our theory also takes variables as the primitive entities in the language, not events which permits us to include "enabling" and "preventing" relationships as part of the mechanism.

model, regardless of the variable ordering. However, following the standard method of scientific induction, it is reasonable to rule out any model for which we find a simpler, less expressive model, equally consistent with the data (see Definition 6). Models that survive this selection are called "minimal models" and with this notion, we can construct our definition of inferred causation:

"A variable X is said to have a causal influence on a variable Y if a strictly directed path from X to Y exists in every minimal model consistent with the data"

Definition 3 Given a set of observable variables $O \subseteq U$, a latent structure is a pair $L = \langle D, O \rangle$ where D is a causal model over U .

Definition 4 One latent structure $L = \langle D, O \rangle$ is preferred to another $L' = \langle D', O \rangle$ (written $L \preceq L'$) iff D' can mimic D over O , i.e. for every Θ_D there exists a $\Theta'_{D'}$ s.t. $P_{[O]}(\langle D', \Theta'_{D'} \rangle) = P_{[O]}(\langle D, \Theta_D \rangle)$

Two latent structures are equivalent, written $L' \equiv L$, iff $L \preceq L'$ and $L \succeq L'$.

Note that the preference for simplicity imposed by Definition 4 is gauged by the expressive power of a model, not by its syntactic description. For example, one latent structure L_1 may invoke many more parameters than L_2 and still be preferred, if L_2 is capable of accommodating a richer set of probability distributions over the observables. One reason scientists prefer simpler models is that such models are more constrained, thus more falsifiable; they provide the scientist with less opportunities to overfit the data hindsightedly and, therefore attain greater credibility [Pearl, 1978, Popper, 1959].

We also note that the set of dependencies induced by a causal model provides a measure of its expressive power, i.e., its power of mimicing other models. Indeed, L_1 cannot be preferred to L_2 if there is even one observable dependency that is induced by L_1 and not by L_2 . Thus, tests for preference and equivalence can often be reduced to tests of induced dependencies which, in turn, can be determined directly from the topology of the dags, without ever concerning ourselves with the set of parameters. (For example, see Theorem 1 below and [Frydenberg, 1989, Pearl et al., 1989, Verma and Pearl, 1990]).

Definition 5 A latent structure L is minimal with respect to a class \mathcal{L} of latent structures iff for every $L' \in \mathcal{L}$, $L \equiv L'$ whenever $L' \preceq L$.

Definition 6 $L = \langle D, O \rangle$ is consistent with a distribution \hat{P} over O if D can accommodate some theory that generates \hat{P} , i.e. there exists a Θ_D s.t. $P_{[O]}(\langle D, \Theta_D \rangle) = \hat{P}$

Clearly, a necessary (and often sufficient) condition for

L to be consistent with \hat{P} , is that the structure of L can account for all the dependencies embodied in \hat{P} .

Definition 7 (Inferred Causation) Given \hat{P} , a variable C has a causal influence on E iff there exists a directed path $C \rightarrow^* E$ in every minimal latent structure consistent with \hat{P} .

We view this definition as normative, because it is based on one of the least disputed norms of scientific investigation: Occam's razor in its semantical casting. However, as with any scientific inquiry, we make no claims that this definition is guaranteed to always identify stable physical mechanisms in nature; it identifies the only mechanisms we can plausibly infer from non-experimental data.

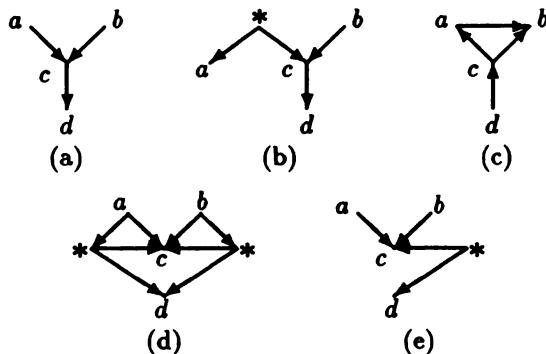


Figure 1: Causal models illustrating the soundness of $a \rightarrow d$. The node (*) represents a hidden variable.

As an example of a causal relation that is identified by the definition above, imagine that observations taken over four variables $\{a, b, c, d\}$ reveal two vanishing dependencies: “ a is independent of b ” and “ d is independent of $\{a, b\}$ given c ”. Assume further that the data reveals no other independence, except those that logically follow from these two. This dependence pattern would be typical for example, of the following variables: $a = \text{having cold}$, $b = \text{having hay-fever}$, $c = \text{having to sneeze}$, $d = \text{having to wipe ones nose}$. It is not hard to see that any model which explains the dependence between c and d by an arrow from d to c , or by a hidden common cause (*) between the two, cannot be minimal, because any such model would be able to out-mimic the one shown in Figure 1(a) which reflects all observed independencies. For example, the model of Figure 1(c), unlike that of Figure 1(a), accommodates distributions with arbitrary relations between a and b . Similarly, Figure 1(d) is not minimal as it fails to impose the conditional independence between d and $\{a, b\}$ given c . In contrast, Figure 1(e) is not consistent with the data since it imposes a marginal independence between $\{a, b\}$ and d , which was not observed.

4 Proof Theory and Stable Distributions

It turns out that while the minimality principle is sufficient for forming a normative and operational theory of causation, it does not guarantee that the search through the vast space of minimal models would be computationally practical. If Nature truly conspires to conceal the structure of the underlying model she could still annotate that model with a distribution that matches many minimal models, having totally disparate structures. To facilitate an effective proof theory, we rule out such eventualities, and impose a restriction on the distribution called “stability” (or “dag-isomorphism” in [Pearl, 1988b]). It conveys the assumption that all vanishing dependencies are structural, not formed by incidental equalities of numerical parameters⁵.

Definition 8 Let $I(P)$ denote the set of all conditional independence relationships embodied in P . A causal theory $T = \langle D, \Theta_D \rangle$ generates a stable distribution iff it contains no extraneous independences, i.e. $I(P(\langle D, \Theta_D \rangle)) \subseteq I(P(\langle D, \Theta'_D \rangle))$ for any set of parameters Θ'_D .

With the added assumption of stability, every distribution has a unique causal model (up to equivalence), as long as there are no hidden variables. This uniqueness follows from the fact the structural constraints that an underlying dag imposes upon the probability distribution are equivalent to a finite set of conditional independence relationships asserting that, given its parents, each variable is conditionally independent of all its non-descendants. Therefore two causal models are equivalent (i.e. they can mimic each other) if and only if they relay the same dependency information. The following theorem, which is founded upon the dependency information, states necessary and sufficient conditions for equivalence of causal models which contain no hidden variables.

Theorem 1 [Verma and Pearl, 1990] When $U = O$, two causal models are equivalent iff their dags have the same links and same set of uncoupled head-to-head nodes⁶.

The search for the minimal model then boils down to recovering the structure of the underlying dag from queries about the dependencies portrayed in that dag. This search is exponential in general, but simplifies significantly when the underlying

⁵It is possible to show that, if the parameters are chosen at random from any reasonable distribution, then any unstable distribution has measure zero [Spirtes et al., 1989]. Stability precludes deterministic constraints. Less restrictive assumptions are treated in [Geiger et al., 1990].

⁶i.e. converging arrows emanating from non-adjacent nodes, such as $a \rightarrow c \leftarrow b$ in Figure 1(a).

structure is sparse (see [Spirtes and Glymour, 1991, Verma and Pearl, 1990] for such algorithms).

Unfortunately, the constraints that a latent structure impose upon the distribution cannot be completely characterized by any set of dependency statements. However, the maximal set of sound constraints can be identified [Verma and Pearl, 1990] and it is this set that permits us to recover sound fragments of latent structures.

5 Recovering Latent Structures

When Nature decides to "hide" some variables, the observed distribution \hat{P} need no longer be stable relative to the observable set O , i.e. \hat{P} may result from many equivalent minimal latent structures, each containing any number of hidden variables. Fortunately, rather than having to search through this unbounded space of latent structures, it turns out that for every latent structure L , there is a dependency-equivalent latent structure called the projection of L on O in which every unobserved node is a root node with exactly two observed children:

Definition 9 A latent structure $L_{[O]} = \langle D_{[O]}, O \rangle$ is a projection of another latent structure L iff

1. Every unobservable variable of $D_{[O]}$ is a parentless common cause of exactly two non-adjacent observable variables.
2. For every stable distribution P generated by L , there exists a stable distribution P' generated by $L_{[O]}$ such that $I(P_{[O]}) = I(P'_{[O]})$.

Theorem 2 Any latent structure has at least one projection (identifiable in linear time).

It is convenient to represent projections by bi-directional graph with only the observed variables as vertices (i.e., leaving the hidden variables implicit). Each bi-directed link in such a graph represents a common hidden cause of the variables corresponding to the link's end points.

Theorem 2 renders our definition of inferred causation (Definition 7) operational; we will show (Theorem 3) that if a certain link exists in a distinguished projection of any minimal model of \hat{P} , it must indicate the existence of a causal path in every minimal model of \hat{P} . Thus the search reduces to finding a projection of any minimal model of \hat{P} and identifying the appropriate links. Remarkably, these links can be identified by a simple procedure, the IC-algorithm, that is not more complex than that which recovers the unique minimal model in the case of fully observable structures.

IC-Algorithm (Inductive Causation)

Input: \hat{P} a sampled distribution.

Output: $\text{core}(\hat{P})$ a marked hybrid acyclic graph.

1. For each pair of variables a and b , search for a set S_{ab} such that (a, S_{ab}, b) is in $I(\hat{P})$, namely a and b are independent in \hat{P} , conditioned on S_{ab} . If there is no such S_{ab} , place an undirected link between the variables.
2. For each pair of non-adjacent variables a and b with a common neighbor c , check if $c \in S_{ab}$. If it is, then continue. If it is not, then add arrowheads pointing at c , (i.e. $a \rightarrow c \leftarrow b$).
3. Form $\text{core}(\hat{P})$ by recursively adding arrowheads according to the following two rules:
 \overline{ab} and there is a strictly directed path from a to b then add an arrowhead at b .
If a and b are not adjacent but \overrightarrow{ac} and $c - b$, then direct the link $c \rightarrow b$.
4. Mark any uni-directed link $a \rightarrow b$ if there is some link with an arrowhead directed at a .

The result of this procedure is a substructure called $\text{core}(\hat{P})$ in which every marked uni-directed arrow $X \rightarrow Y$ stands for the statement: "X has a causal influence on Y (in all minimal latent structures consistent with the data)". We call these relationships "genuine" causal influences (e.g. $c \rightarrow d$ in previous Figure 1).

Definition 10 For any latent structure L , $\text{core}(L)$ is defined as the hybrid graph⁸ satisfying (1) two nodes are adjacent in $\text{core}(L)$ iff they are adjacent or they have a common unobserved cause in every projection of L , and (2) a link between a and b has an arrowhead pointing at b iff $a \rightarrow b$ or a and b have a common unobserved cause in every projection of L .

Theorem 3 For any latent structure $L = \langle D, O \rangle$ and an associated theory $T = \langle D, \Theta_D \rangle$ if $P(T)$ is stable then $\text{core}(L) = \text{core}(P_{[O]}(T))$.

Corollary 1 If every link of the directed path $C \rightarrow^* E$ is marked in $\text{core}(\hat{P})$ then C has a causal influence on E according to \hat{P} .

6 Probabilistic Definitions for Causal Relations

The IC-algorithm takes a distribution \hat{P} and outputs a dag, some of its links are marked uni-directional

⁷ \overline{ab} denotes adjacency, i.e. $a - b$, $a \rightarrow b$, $a \leftarrow b$ or $a \leftrightarrow b$;
⁸ \rightarrow ab denotes either $a \rightarrow b$ or $a \leftrightarrow b$.

⁹In a hybrid graph links may be undirected, uni-directed or bi-directed.

(denoting genuine causation), some are unmarked unidirectional (denoting potential causation), some are bidirectional (denoting spurious association) and some are undirected (denoting relationships that remain undetermined). The conditions which give rise to these labelings constitute operational definitions for the various kinds of causal relationships. In this section we present explicit definitions of potential and genuine causation, as they emerge from Theorem 3 and the IC-algorithm. Note that in all these definitions, the criterion for causation between two variables, X and Y , will require that a third variable Z exhibit a specific pattern of interactions with X and Y . This is not surprising, since the very essence of causal claims is to stipulate the behavior of X and Y under the influence of a third variable, one that corresponds to an external control of X . Therefore, our definitions are in line with the paradigm of “no causation without manipulation” [Holland, 1986]). The difference is only that the variable Z , acting as a virtual control of X , must be identified within the data itself. The IC-algorithm provides a systematic way of searching for variables Z that qualify as virtual controls.

Detailed discussions of these definitions in terms of virtual control are given in Sections 7 and 8.

Definition 11 (Potential Cause) A variable X has a potential causal influence on another variable Y (inferable from \hat{P}), if

1. X and Y are dependent in every context.
2. There exists a variable Z and a context S such that
 - (i) X and Z are independent given S
 - (ii) Z and Y are dependent given S

Note that this definition precludes a variable X from being a potential cause of itself or of any other variable which functionally determines X .

Definition 12 (Genuine Cause) A variable X has a genuine causal influence on another variable Y if there exists a variable Z such that either:

1. X is a potential cause of Y and there exists a context S satisfying:
 - (i) Z is a potential cause of X
 - (ii) Z and Y are dependent given S .
 - (iii) Z and Y are independent given $S \cup X$,
 or,
2. X is a genuine cause of Z and Z is a genuine cause of Y .

Definition 13 (Spurious Association) Two variables X and Y are spuriously associated if they are

dependent in some context S and there exists two other variables Z_1 and Z_2 such that:

1. Z_1 and X are dependent given S
2. Z_2 and Y are dependent given S
3. Z_1 and Y are independent given S
4. Z_2 and X are independent given S

Succinctly, using the predicates I and $\neg I$ to denote independence and dependence respectively, the conditions above can be written:

1. $\neg I(Z_1, X|S)$
2. $\neg I(Z_2, Y|S)$
3. $I(Z_1, Y|S)$
4. $I(Z_2, X|S)$

Definition 11 was formulated in [Pearl, 1990] as a relation between events (rather than variables) with the added condition $P(Y|X) > P(Y)$ in the spirit of [Good, 1983, Reichenbach, 1956, Suppes, 1970]. Condition 1 in Definition 12 may be established either by statistical methods (per Definition 11) or by other sources of information e.g., experimental studies or temporal succession (i.e. that Z precedes X in time).

When temporal information is available, as it is assumed in the most theories of causality ([Granger, 1988, Spohn, 1983, Suppes, 1970]), then Definitions 12 and 13 simplify considerably because every variable preceding and adjacent to X now qualifies as a “potential cause” of X . Moreover, adjacency (i.e. condition 1 of Definition 11) is not required as long as the context S is confined to be earlier than S . These considerations lead to simpler conditions distinguishing genuine from spurious causes as shown next.

Definition 14 (Genuine Causation with temporal information) A variable X has a causal influence on Y if there is a third variable Z and a context S , both occurring before X such that:

1. $\neg I(Z, Y|S)$
2. $I(Z, Y|S \cup X)$

Definition 15 (Spurious Association with temporal information) Two variables X and Y are spuriously associated if they are dependent in some context S , X precedes Y and there exists a variable Z satisfying:

1. $I(Z, Y|S)$
2. $\neg I(Z, X|S)$

7 Causal Intuition and Virtual Experiments

This section explains how the formulation introduced above conforms to common intuition about causation and, in particular, how symmetric probabilistic dependencies can be transformed into judgements about causal influences. We shall first uncover the intuition behind Definition 14, assuming the availability of temporal information, then (in Section 8) generalize to non-temporal data, per Definition 12.

The common intuition about causation is captured by the heuristic definition [Rubin, 1989]: “ X is a cause for Y if an external agent interfering only with X can affect Y ”.

Thus, causal claims are much bolder than those made by probability statements; not only do they summarize relationships that hold in the distribution underlying the data, but they also predict relationships that should hold when the distribution undergoes changes, such as those inferable from external intervention. The claim “ X causes Y ” asserts the existence of a *stable* dependence between X and Y , one that cannot be attributed to some prior cause common to both, and one that should be preserved when an exogenous control is applied to X .

This intuition requires the formalization of three notions:

1. That the intervening agent be “external” (or “exogenous”)
2. That the agent can “affect” Y
3. That the agent interferes “only” with X

If we label the behavior of the intervening agent by a variable Z , then these notions can be given the following probabilistic explications:

1. **Externality of Z :** Variations in Z must be independent of any factors W which precede X , i.e.,

$$I(Z, W) \quad \forall W : t_W < t_X \quad (1)$$

2. **Control:** For Z to effect changes in Y (via X) we require that Z and Y be dependent, written:

$$\neg I(Z, Y) \quad (2)$$

3. **Locality:** To ensure that Z interferes “only” with X , i.e., that its entire effect on Y is mediated by X , we use the conditional independence assertion:

$$I(Z, Y | X) \quad (3)$$

to read “ Z is independent of Y , given X ”.

Note that (1) and (2) imply (by the axioms of conditional independence [Pearl, 1988b]) that X and Y are dependent, namely, $\neg I(X, Y)$.

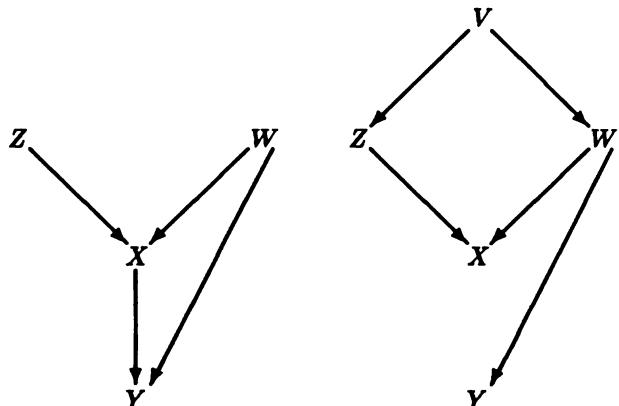


Figure 2

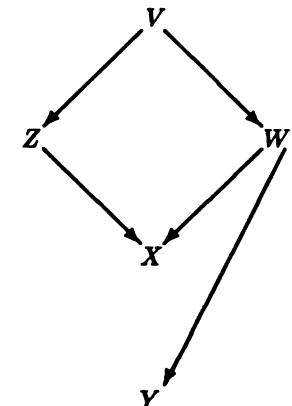


Figure 3

Conditions (1) through (3) constitute the traditional premises behind controlled statistical experiments, with (1) reflecting the requirement that units selected for the experiment be chosen at random from the population under study. They guarantee that any dependency observed between X and Y cannot be explained away by holding fixed some factor W preceding X , hence it must be attributed to genuine causation. The sufficiency of these premises is clearly not a theorem of probability theory, as it relies on temporal relationships among the variables. However, it can be derived from probability theory together with Reichenbach’s principle [Reichenbach, 1956], stating that every dependence $\neg I(X, Y)$ requires a causal explanation, namely either one of the variables causes the other, or there must be a variable W preceding X and Y such that $I(X, Y | W)$ (see Figure 2). Indeed, if there is no back path from Z to Y through W (Eq. (1)) and no direct path from Z to Y avoiding X (Eq. (3)) then there must be a causal path from X to Y that is responsible for the dependence in Eq. (2)⁹.

In non-experimental situations it is not practical to detach X completely from its natural surrounding and to subject it to the exclusive control of an exogenous (and randomized) variable Z . Instead, one could view some of X ’s natural causes as “virtual controls” and, provided certain conditions are met, use the latter to reveal non-spurious causal relationship between X and Y . In so doing we compromise, of course, condition (1), because we can no longer guarantee that those natural causes of X are not themselves affected by other causes which, in turn, might influence Y (see Figure 3). However, it turns out that for stable distributions, conditions (2) and (3) are sufficient to guarantee that the association between X and Y is non-spurious, thus justifying Definition 14 for genuine causation.

The intuition goes as follows (see Figure 3): If the de-

⁹Cartwright [Cartwright, 1989] offers a sufficiency proof in the context of linear models.

pendency between Z and Y (and similarly, between X and Y) is spurious, namely, X and Y are merely manifestations of some common cause W , there is no reason then for X to screen-off Y from Z , and condition (2) should be violated. In case condition (2) is accidentally satisfied by some strange combination of parameters, it is bound to be “unstable”, as it will be perturbed with any slight change of experimental conditions.

Conditions (2) and (3) are identical to those in Definition 14, save for the context S which is common to both. The inclusion of the fixed context S is legitimized by noting that if $P(X, Y, Z)$ is a marginal of a stable distribution, then so is the conditional distribution $P(X, Y, Z|S = s)$, as long as S corresponds to variables which precede X .

Definition 14 constitutes an alternative way of recovering causal structures, more flexible than the IC-algorithm; we search the data for three variables Z, X, Y (in this temporal order) that satisfy the two conditions in some context $S = s$, and when such a triple is found, X is proclaimed to have a genuine causal influence on Y . Clearly, permitting an arbitrary context S increases the number of genuine causal influences that can be identified in any given data; marginal independencies and even 1-place conditional independencies are rare phenomenon.

Note that failing to satisfy the test for genuine causation does not mean that such relationship is necessarily absent between the quantities under study. Rather, it means that the data available cannot substantiate the claim of genuine causation. To further test such claims one may need to either conduct experimental studies, or consult a richer data set where virtual control variables are found.

In testing this modeling scheme on real life data, we have examined the observations reported in Sewall Wright's seminal paper “Corn and Hog Correlations” [Wright, 1925]. As expected, corn-price (X) can clearly be identified as a cause of hog-price (Y), not the other way around. The reason lies in the existence of the variable corn-crop (Z) that, by satisfying the conditions of Definition 14 (with $S = \emptyset$), acts as a virtual control of X (see Figure 2). To test for the possibility of reciprocal causation, one can try to find a virtual controller for Y , for example, the amount of hog-breeding (Z'). However, it turns out that Z' is not screened off from X by Y (possibly because corn prices exert direct influence over farmer's decision to breed more hogs), hence, failing condition 3, Y disqualifies as a genuine cause of X . Such distinctions are important to policy makers in deciding, for example, which commodity, corn or hog, should be subsidized or taxed.

8 Non-Temporal Causation and Statistical Time

When temporal information is unavailable the condition that Z precede X (Definition 14) cannot be tested directly and must be replaced by an equivalent condition, based on dependence information. As it turns out, the only reason we had to require that Z precede X is to rule out the possibility that Z is a causal consequence of X ; if it were a consequence of X then the dependency between Z and Y could easily be explained away be a common cause W of X and Y (see Figure 2).

The information that permits us to conclude that one variable is not a causal consequence of another comes in the form of an “intransitive triplet”, such as the variables a, b and c in Figure 1(a) satisfying: $I(a, b)$, $\neg I(a, c)$ and $\neg I(b, c)$. The argument goes as follows: If we create conditions (fixing S_{ab}) where two variables, a and b , are each correlated with a third variable c but are independent of each other, then the third variable cannot act as a cause of a or b , (recall that in stable distributions, common causes induce dependence among their effects); it must be either their common effect, $a \rightarrow c \leftarrow b$, or be associated with a and b via common causes, forming a pattern such as $a \leftrightarrow c \leftrightarrow b$. This is indeed the eventuality that permits our algorithm to begin orienting edges in the graph (step 2), and assign arrowheads pointing at c . It is also this intransitive pattern which is used to ensure that X is not a consequence of Y (in Definition 11) and that Z is not a consequence of X (in Definition 12). In definition 14 we have two intransitive triplets, (Z_1, X, Y) and (X, Y, Z_2) , thus ruling out direct causal influence between X and Y , implying spurious associations as the only explanation for their dependence.

This interpretation of the intransitive triple is in line with the “virtual control” view of causation. For example, one of the reasons people insist that the rain causes the grass to become wet, and not the other way around, is that they can find other means of getting the grass wet, totally independent of the rain. Transferred to our chain $a - c - b$, we can preclude c from being a cause of a if we find another means (b) of potentially controlling c without affecting a [Pearl, 1988a, p. 396].

Determining the direction of causal influences from nontemporal data raises some interesting philosophical questions about the nature of time and causal explanations. For example, can the orientation assigned to the arrow $X \rightarrow Y$ in Definition 14 ever clash with temporal information (say by a subsequent discovery that Y precedes X)? Alternatively, since the rationale behind Definition 14 is based on strong intuitions about how causal influences should behave (statistically), it is apparent that such clashes, if they occur, are rather rare. The question arises then, why? Why should orientations determined solely by statistical dependencies

have anything to do with the flow of time?

In human discourse, causal explanations indeed carry two connotations, temporal and statistical. The temporal aspect is represented by the convention that a cause should precede its effect. The statistical aspect expects causal explanations (once accounted for) to screen off their effects, i.e., render their effects conditionally independent¹⁰. More generally, causal explanations are expected to obey many of the rules that govern paths in a directed acyclic graphs (e.g., the intransitive triplet criterion for potential causation, Section 7). This leads to the observation that, if agreement is to hold between the temporal and statistical aspects of causation, natural statistical phenomena must exhibit some basic temporal bias. Indeed, we often encounter phenomenon where knowledge of a present state renders the variables of the future state conditionally independent (e.g., multi-variables economic time series as in Eq. (4) below). We rarely find the converse phenomenon, where knowledge of the present state would render the components of the past state conditionally independent. The question arises whether there is any compelling reason for this temporal bias.

A convenient way to articulate this bias is through the notion of “Statistical Time”.

Definition 16 (Statistical Time) *Given an empirical distribution P , a statistical time of P is any ordering of the variables that agrees with at least one minimal causal model consistent with P .*

We see, for example, that a scalar Markov-chain process has many statistical times; one coinciding with the physical time, one opposite to it and the others correspond to any time ordering of the variables away from some chosen variable. On the other hand a process governed by two coupled Markov chains,

$$\begin{aligned} X_t &= \alpha X_{t-1} + \beta Y_{t-1} + \xi_t \\ Y_t &= \gamma X_{t-1} + \delta Y_{t-1} + \xi'_t, \end{aligned} \quad (4)$$

has only one statistical time – the one coinciding with

¹⁰This principle, known as Reichenbach’s “conjunctive fork” or “common-cause” criterion [Reichenbach, 1956, Suppes and Zaniotti, 1981] has been criticized by Salmon [Salmon, 1984], who showed that some events would qualify as causal explanations though they fail to meet Reichenbach’s criterion. Salmon admits, however, that when a conjunctive fork does occur, the screening off variable is expected to be the cause of the other two, not the effect [Salmon, 1984, p. 167]. He notes that it is difficult to find physically meaningful examples where a response variable renders its two causes conditionally independent (although this would not violate any axiom of probability theory). This asymmetry is further evidence that humans tend to reject causal theories that yield unstable distributions.

the physical time¹¹. Indeed, running the IC-algorithm on samples taken from such a process, while suppressing all temporal information, quickly identifies the components of X_{t-1} and Y_{t-1} as genuine causes of X_t and Y_t . This can be seen from Definition 11, where X_{t-2} qualifies as a potential cause of X_{t-1} using $Z = Y_{t-2}$ and $S = \{X_{t-3}, Y_{t-3}\}$, and Definition 12, where X_{t-1} qualifies as a genuine cause of X_t using $Z = X_{t-2}$ and $S = \{Y_{t-1}\}$ of X_t .

The temporal bias postulated earlier can be expressed as follows:

Conjecture 1 (Temporal Bias) *In most natural phenomenon, the physical time coincides with at least one statistical time.*

Reichenbach [Reichenbach, 1956] attributed the asymmetry associated with his conjunctive fork to the second law of thermodynamics. We are not sure at this point whether the second law can provide a full account of the temporal bias as defined above, since the influence of the external noise ξ_t and ξ'_t renders the process in (4) nonconservative¹². What is clear, however, is that the temporal bias is *language dependent*. For example, expressing Eq.(4) in a different coordinate system (say, using a unitary transformation $(X'_t, Y'_t) = U(X_t, Y_t)$), it is possible to make the statistical time (in the (X', Y') representation) run contrary to the physical time. This suggests that the apparent agreement between the physical and statistical times is a byproduct of human choice of linguistic primitives and, moreover, that the choice is compelled by a survival pressure to facilitate predictions at the expense of diagnosis and planning.

9 Conclusions

The theory presented in this paper should dispel the belief that statistical analysis can never distinguish genuine causation from spurious covariation. This belief, shaped and nurtured by generations of statisticians [Fisher, 1953, Keynes, 1939, Ling, 1983, Niles, 1921] has been a major hindrance in the way of developing a satisfactory, non-circular account of causation. In the words of Gardenfors [Gardenfors, 1988, page 193]:

In order to distinguish genuine from spurious causes, we must already know the causally relevant background factors. ... Further, the extra amount of information is substantial: In order to determine whether C is a cause of E, *all* causally relevant background factors must be available. It seems clear that we

¹¹ ξ_t and ξ'_t are assumed to be two independent, white noise time series. Also $\alpha \neq \delta$ and $\gamma \neq \beta$.

¹²We are grateful to Seth Lloyd for this observation.

often have determinate beliefs about causal relations between events, even if we do not know exactly which factors are causally relevant to the events in question¹³.

This paper shows that such extra information is often unnecessary: Under the assumptions of model-minimality (and/or stability), there are patterns of dependencies that should be sufficient to uncover genuine causal relationships. These relationships cannot be attributed to hidden causes lest we violate one of the basic maxims of scientific methodology: the semantical version of Occam's razor. Adherence to this maxim explains why humans reach consensus regarding the directionality and nonspuriousness of causal relationships, in the face of opposing alternatives, perfectly consistent with experience. Echoing Cartwright [Cartwright, 1989] we summarize our claim with the slogan "No Causes In, Some Causes Out".

From a methodological viewpoint, our theory should settle some of the on going disputes regarding the validity of path-analytic approaches to causal modeling in the social sciences [Freedman, 1987, Ling, 1983]. It shows that the basic philosophy governing path-analytic methods is legitimate, faithfully adhering to the traditional norms of scientific investigation. At the same time our results also explicate the assumptions upon which these methods are based, and the conditions that must be fulfilled before claims made by these methods can be accepted. Specifically, our analysis makes it clear that causal modeling must begin with *vanishing (conditional) dependencies* (i.e. missing links in their graphical representations). Models that embody no vanishing dependencies contain no virtual control variables, hence, the causal component of their claims cannot be substantiated by observational studies. With such models, the data can be used only for estimating the parameters of the causal links once we are absolutely sure of the causal structure, but the structure itself, and especially the directionality of the links, cannot be inferred from the data. Unfortunately, such models are often employed in the social and behavioral sciences e.g. [Kenny, 1979].

On the practical side, we have shown that the assumption of model minimality, together with that of "stability" (no accidental independencies) lead to an effective algorithm of recovering causal structures, transparent as well as latent. Simulation studies conducted at our laboratory show that networks containing tens of variables require less than 5000 samples to have their structure recovered by the algorithm. For example, 1000 samples taken from the process shown in Eq. (5), each containing ten successive X,Y pairs, were sufficient for recovering its double-chain structure (and the correct direction of time). The greater the noise,

the quicker the recovery.

Another result of practical importance is the following: Given a proposed causal theory of some phenomenon, our algorithm can identify in linear time those causal relationships that could potentially be substantiated by observational studies, and those whose directionality non-spuriousness can only be determined by controlled, manipulative experiments.

It should also be interesting to explore how the new criteria for causation could benefit current research in machine learning. In some sense, our method resembles a search through elements of a version space [Mitchell, 1982], where each hypothesis stands for a causal theory. Unfortunately, this is where the resemblance ends. The prevailing paradigm in the machine learning literature has been to define each hypothesis (or theory, or concept) as a subset of observable instances; once we observe the entire extension of this subset, the hypothesis is defined unambiguously. This is not the case in causal modeling. Even if the training sample exhausts the hypothesis subset (in our case, this corresponds to observing P precisely), we are still left with a vast number of equivalent causal theories, each stipulating a drastically different set of causal claims. Fitness to data, therefore, is an insufficient criterion for validating causal theories. Whereas in traditional learning tasks we attempt to generalize from one set of instances to another, the causal modeling task is to generalize from behavior under one set of conditions to behavior under another set. Causal models should therefore be chosen by a criterion that challenges their stability against changing conditions, and these show up in the data in the form of virtual control variables. Thus, the dependence patterns identified by definition 11 through 14 constitute islands of stability as well as virtual validation tests for causal models. It would be interesting to examine whether these criteria, when incorporated into existing machine learning programs would improve the stability of theories discovered by such programs.

Acknowledgement

We are grateful to Clark Glymour for posing the problem of equivalence in latent structures. Some of the problems treated in this paper were independently explored by Glymour, Spirtes and Schienes [Spirtes et al., 1989, Spirtes and Glymour, 1991], and we thank them for sharing this information with us. Discussions and correspondence with P. Bentler, D. Geiger, C. Granger, M. Hanssens, J. de Leeuw, S. Lloyd, R. Otte, A. Paz, B. Skyrms and P. Suppes are greatly appreciated.

References

[Bobrow, 1985] Bobrow, D. (1985). *Qualitative Re-*

¹³See also Cartwright [Cartwright, 1989] for a similar position, and for a survey of the literature.

- soning about Physical Systems.* MIT Press, Cambridge, MA.
- [Cartwright, 1989] Cartwright, N. (1989). *Nature Capacities and Their Measurements*. Clarendon Press, Oxford.
- [Cliff, 1983] Cliff, N. (1983). Some cautions concerning the application of causal modeling methods. *Multivariate behavioral research*, 18:115 – 126.
- [de Kleer and Brown, 1986] de Kleer, J. and Brown, J. S. (1986). Theories of causal ordering. *Artificial Intelligence*, 29(1):33 – 62.
- [Dechter and Pearl, 1990] Dechter, R. and Pearl, J. (1990). Directional constraint networks: A relational framework for causal modeling. Technical Report R-153, UCLA Cognitive Systems Laboratory.
- [Eells and Sober, 1983] Eells, E. and Sober, E. (1983). Probabilistic causality. *Philosophy of Science*, 50:35 – 57.
- [Fisher, 1953] Fisher, R. A. (1953). *Design of Experiments*. Oliver and Boyd, London.
- [Forbus and Gentner, 1986] Forbus, K. D. and Gentner, D. (1986). Causal reasoning about quantities. *Proceedings Cognitive Science Society*, pages 196 – 207.
- [Freedman, 1987] Freedman, D. (1987). As others see us: A case study in path analysis (with discussion). *Journal of Educational Statistics*, 12:101 – 223.
- [Frydenberg, 1989] Frydenberg, M. (1989). The chain graph markov property. Technical Report 186, Department of Theoretical Statistics, University of Aarhus, Denmark.
- [Gardenfors, 1988] Gardenfors, P. (1988). Causation and the dynamics of belief. In Harper, W. and Skyrms, B., editors, *Causation in Decision, Belief Change and Statistics II*, pages 85 – 104. Kluwer Academic Publishers.
- [Geffner, 1989] Geffner, H. (1989). *Default Reasoning: Causal and Conditional Theories*. PhD thesis, UCLA Computer Science Department, Los Angeles, CA.
- [Geiger et al., 1990] Geiger, D., Paz, A., and Pearl, J. (1990). Learning causal trees from dependence information. In *Proceedings, AAAI-90*, pages 770 – 776, Boston, MA.
- [Glymour et al., 1987] Glymour, C., Scheines, R., Spirtes, P., and Kelly, K. (1987). *Discovering Causal Structure*. Academic Press, New York.
- [Good, 1983] Good, I. J. (1983). A causal calculus. *British Journal for Philosophy of Science*, 11 and 12 and 13:305 – 328 and 43 – 51 and 88. reprinted as Ch. 21 in Good Thinking University of Minnesota Press, Minneapolis, MN.
- [Granger, 1988] Granger, C. W. J. (1988). Causality testing in a decision science. In Harper, W. and Skyrms, B., editors, *Causation in Decision, Belief Change and Statistics I*, pages 1 – 20. Kluwer Academic Publishers.
- [Holland, 1986] Holland, P. (1986). Statistics and causal inference. *Journal of the American Statistical Association*, 81:945 – 960.
- [Iwasaki and Simon, 1986] Iwasaki, Y. and Simon, H. A. (1986). Causality in device behavior. *Artificial Intelligence*, 29(1):3 – 32.
- [Kautz, 1987] Kautz, H. (1987). *A formal Theory of Plan Recognition*. PhD thesis, University of Rochester, Rochester, N.Y.
- [Kenny, 1979] Kenny, D. A. (1979). *Correlation and Causality*. Wiley, New York.
- [Keynes, 1939] Keynes, J. M. (1939). Professor tinbergen's method. *Economic Journal*, 49:560.
- [Lifschitz, 1987] Lifschitz, V. (1987). Formal theories of action. In *Workshop of the Frame Problem in AI*, pages 35 – 57, Kansas.
- [Ling, 1983] Ling, R. (1983). Review of "Correlation and Causation" by D. Kenny. *Journal of the American Statistical Association*, pages 489 – 491.
- [Mitchell, 1982] Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, 18:203 – 226.
- [Niles, 1921] Niles, H. E. (1921). Correlation, causation, and Wright theory of "path coefficients". *Genetics*, 7:258 – 273.
- [Patil et al., 1982] Patil, R. S., Szolovitz, P., and Schwartz, W. B. (1982). Causal understanding of patient illness in patient diagnosis. In *Proceedings of AAAI-82*, pages 345 – 348.
- [Pearl, 1978] Pearl, J. (1978). On the connection between the complexity and credibility of inferred models. *International Journal of General Systems*, 4:255 – 264.
- [Pearl, 1988a] Pearl, J. (1988a). Embracing causality in formal reasoning. *Artificial Intelligence*, 35(2):259 – 71.
- [Pearl, 1988b] Pearl, J. (1988b). *Probabilistic Reasoning in Intelligent Systems*. Morgan-Kaufman, San Mateo, CA.
- [Pearl, 1990] Pearl, J. (1990). Probabilistic and qualitative abduction. In *Proceedings of AAAI Spring Symposium on Abduction*, pages 155 – 158, Stanford.
- [Pearl et al., 1989] Pearl, J., Geiger, D., and Verma, T. S. (1989). The logic of influence diagrams. In Oliver, R. M. and Smith, J. Q., editors, *Influence Diagrams, Belief Networks and Decision Analysis*, pages 67 – 87. John Wiley and Sons, Ltd., Sussex, England.
- [Popper, 1959] Popper, K. R. (1959). *The Logic of Scientific Discovery*. Basic Books, New York.

- [Reichenbach, 1956] Reichenbach, H. (1956). *The Direction of Time*. University of California Press, Berkeley.
- [Reiter, 1987] Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57 – 95.
- [Rubin, 1989] Rubin, H. (1989). Discussion of “The Logic of Influence Diagrams” by Pearl et al. In Oliver, R. M. and Smith, J. Q., editors, *Influence Diagrams, Belief Networks and Decision Analysis*, pages 83 – 85. John Wiley and Sons, Ltd., Sussex, England.
- [Salmon, 1984] Salmon, W. (1984). *Scientific explanation and the causal structure of the world*. Princeton University Press., Princeton.
- [Shoham, 1988] Shoham, Y. (1988). *Reasoning About Change*. MIT Press, Boston, MA.
- [Simon, 1954] Simon, H. (1954). Spurious correlations: A causal interpretation. *Journal American Statistical Association*, 49:469 – 492.
- [Skyrms, 1986] Skyrms, B. (1986). *Causal Necessity*. Yale University Press, New Haven, CT.
- [Spirtes and Glymour, 1991] Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9.
- [Spirtes et al., 1989] Spirtes, P., Glymour, C., and Scheines, R. (1989). Causality from probability. Technical Report CMU-LCL-89-4, Department of Philosophy Carnegie-Mellon University.
- [Spohn, 1983] Spohn, W. (1983). Deterministic and probabilistic reasons and causes. *Erkenntnis*, 19:371 – 396.
- [Suppes, 1970] Suppes, P. (1970). *A Probabilistic Theory of Causation*. North Holland, Amsterdam.
- [Suppes and Zaniotti, 1981] Suppes, P. and Zaniotti, M. (1981). When are probabilistic explanations possible? *Synthese*, 48:191 – 199.
- [Verma, 1991] Verma, T. S. (1991). Invariant properties of causal models. Technical report, UCLA Cognitive Systems Laboratory.
- [Verma and Pearl, 1990] Verma, T. S. and Pearl, J. (1990). Equivalence and synthesis of causal models. In *Proceedings 6th Conference on Uncertainty in AI*, pages 220 – 227.
- [Wilensky, 1983] Wilensky, R. (1983). *Planning and understanding*. Addison Wesley.
- [Wright, 1925] Wright, S. (1925). Corn and hog correlations. Technical Report 1300, U.S. Department of Agriculture.

The Logic of Epistemic Inconsistency

Tarcisio Pequeno¹ Arthur Buchsbaum

Departamento de Informática
Pontifícia Universidade Católica
Rua Marquês de São Vicente 225
22453 Rio de Janeiro - Brasil

Abstract

The notion of *epistemic inconsistency*, referring to contradictory views about a same situation, is introduced. These contradictions reflect not an anomalous behavior of the state of affairs but the incompleteness (or vagueness) of our knowledge about it. The association of this phenomenon with nonmonotonic reasoning is discussed. A logic, calculus and semantics, aiming to make precise this notion and to enable reasoning on these inconsistent views, without triviality, is presented.

1 INTRODUCTION

Deductive reasoning, our praised paradigm for correct, impeccable, contradiction free thinking, unfortunately has its application restricted to (ideal) situations where complete knowledge about the facts and their relations is available. For more realistic settings, the ones in daily life and in many A.I. applications, reasoning methods allowing the use of super deductive inference rules, such as reasoning by default, for instance, are required. Strictly speaking, these inference rules are not sound. It might happen situations in which the premises are true but not the conclusion. As a consequence of this lack of soundness contradiction may eventually be achieved.

In [Pequeno 1990] it is argued that inconsistency is just a natural companion to nonmonotonic methods of

reasoning and that paraconsistency (the property of a logic admitting non trivial inconsistent theories) should play a role in the formalization of these methods. The argument can be briefly stated as follows.

Nonmonotonic reasoning is applied to situations in which the knowledge is necessarily incomplete, eventually inaccurate as well, and very often involving informations giving evidence to contradictory conclusions. Unlike deduction, this kind of reasoning cannot be performed on local basis, without appealing to context. In the course of reasoning the arguments interfere with each other, generating conflicts and promoting the defeat of partial conclusions.

Furthermore, there is no guarantee that every arising conflict can be resolved. It may perfectly happen two opposite partial conclusions having equal rights to be achieved or, even if there is not such a perfect symmetry, it can happen anyway the available knowledge not enabling a clear decision in favour of one of the alternatives. Thus contradiction arises. In case of deduction this would carry out a revision of premisses by the application of *reductio ad absurdum*. This is not the case here. There is no point in applying *reductio ad absurdum* to contradiction among defeasible conclusions.

In [Pequeno 1990] it is suggested that these contradictory conclusions should be assimilated in a single theory and reasoned out just as any other. This would emphasize the need for a better understanding of this kind of situations in order to provide a purely logical analysis for them. In other words, to achieve these contradictions that emerge in the course of reasoning is just to give the right account for the situation. Obviously this could not be done in classical logic. A special logic, a paraconsistent one, would be required.

¹ On leave from Universidade Federal do Ceará.

So, this is how nonmonotonic reasoning can lead to the adoption of paraconsistency. On the other hand, nonmonotonicity confers dynamics to paraconsistent reasoning. It allows incoming informations eventually to remove contradictions and/or include new ones.

The contamination of nonmonotonic reasoning with unresolvable contradiction is a well known phenomenon. Examples that illustrate it, such as Nixon's diamond, have been recurrent in the literature on nonmonotonic logics. In spite of this recognition, there is an (understandable) resistance in assuming the inconsistent theories that these contradictions seem to imply. One approach to this problem has been simply to avoid the contradiction by dismissing both opposite conclusions. Another approach, taken by Reiter in his default logic, [Reiter 1980], has been to split contradictory conclusions into multiple extensions, each one internally consistent. We are more concerned here with the discussion of this second approach.

The splitting out of diverging default conclusions into multiple extensions has the effect of precluding the purely logical analysis of the whole situation. The contribution of extralogical mechanisms to deal with extensions, in order to perform reasoning, would be required. Furthermore, this approach has an undesired side effect which prevents default logic to avoid unintended extensions (and conclusions) in situations such as the famous "Yale shooting problem", discussed in [Hanks & McDermott 1987].

Consider the following example, taken from [Morris 1988]:

- Animals usually cannot fly;
- Winged animals are exception to this, they can fly;
- Birds are animals;
- Birds normally have wings.

This can be axiomatized, using Reiter's default, as follows:

1. $\text{an}(x)$: $\neg \text{fly}(x) \wedge \neg \text{wing}(x)$
 $\quad \quad \quad \neg \text{fly}(x)$
2. $\text{wing}(x) \rightarrow \text{fly}(x)$
3. $\text{bird}(x) \rightarrow \text{an}(x)$
4. $\text{bird}(x)$: $\text{wing}(x)$
 $\quad \quad \quad \text{wing}(x)$

The following reasoning will then be possible: given that Tweety is a bird, it follows from (3) that it is an animal, and from this and rule (1) that it cannot fly. By *modus tollens* on (2), if it cannot fly it is not winged. With Reiter's default logic this last conclusion prevents the application of the last default rule and therefore, from the single fact that the poor Tweety is a bird, it comes out this bizarre conclusion that it is wingless.

What happened so wrong here? The splitting into two extensions: one in which Tweety is winged and another in which it is not, didn't allow the reasoning to see that, by being a bird, therefore winged, Tweety constitutes an exception to rule (1), which makes this rule not being applicable. This information belonged to another extension and thus could not be seen from the unintended extension. Thus, the dissolving of conflicts by the splitting into extensions prevented the consideration of a relevant piece of evidence, causing the trouble of not handling properly the exception condition.

In [Pequeno 1990] it is presented a logic, the *Inconsistent Default Logic*, IDL in short, which benefits from a tolerant disposition towards contradiction, being able to solve this problem. A general IDL default rule reads as follows:

$$\begin{array}{c} \underline{\text{A} : \text{B} ; \text{C}} \\ \text{B?} \end{array}$$

A is the *antecedent* of the rule and B its *default condition*. C is a *proviso* (its negation is an *exception condition* for the application of the rule). Finally, B? is the *consequent*.

This rule is a modification of Reiter's rule in accordance with the following considerations:

- a defeasible conclusion can never have the same epistemic status as an irrefutable one, obtained from deduction. Thus in IDL the former is distinguished from the later by the use of an interrogation mark (?) suffixing defeasible formulas.

- IDL implements the idea of accommodating conflicting views in a same extension. Therefore, in IDL the defeasible negation of a *default condition* ($\neg \text{B}?$) (we call it a *weak contradiction*) does not prevent the application of the default rule. In order to defeat a default application a *strong contradiction* $\neg \text{B}$ is required. Nixon's example, for instance, when treated in IDL, generates just one extension, containing " $\text{pacifist}(\text{Nixon})?$ " and " $\neg \text{pacifist}(\text{Nixon})?$ ".

- The *seminormal* part of a default rule is frequently used to express an exception condition. In IDL, C is really taken as a *proviso* for the application of the rule, receiving a differentiate treatment. In order to defeat the application of an IDL default rule by its proviso, a weak contradiction, ($\neg \text{C}?$), suffices.

We are now able to see how IDL works in Morris's example. The same argument could be constructed as before up to the temporary conclusion " $\neg \text{winged}(\text{Tweety})?$ ". But now, this does not defeat the application of rule (4) and thus " $\text{winged}(\text{Tweety})?$ " is also achieved. This last conclusion, even being defeasible, is able to prevent the application of rule (1) (by means of its proviso). Therefore " $\neg \text{fly}(\text{Tweety})?$ " is withdrawn together with " $\neg \text{winged}(\text{Tweety})?$ ". So, with IDL, only

the expected conclusions that by being a bird Tweety is winged and can fly are obtained.

We leave to the reader to check it out that IDL would do equally well in the Yale shooting problem (lazy readers have the option to see it in [Pequeno 1990]).

We call *epistemic inconsistency* to inconsistencies in descriptions of a state of affairs reflecting not an inconsistency in the state of affairs itself but a lack in our knowledge about it. This term stands in opposition to *ontological inconsistency*, which refers to an inconsistent behavior of the reality itself (whatever this means) and has been used before, roughly with the same meaning as here, by [Rescher & Brandom 1980].

The Logic of Epistemic Inconsistency, LEI for short, has been designed aiming to make precise this notion. It is intended to reason out meaningfully contradictions resulting from reasoned out incomplete knowledge. Although to serve as the monotonic basis for IDL has been the main motivation for designing LEI, it has an independent existence and an interest in its own. The occurrence of inconsistencies arising from lack of knowledge is not restricted to nonmonotonic or inductive reasoning but, quite on the contrary, it is a very widespread phenomenon. It will be shown that some basic intuitions, reflected in the semantics of LEI, underlie many instances of it.

On the other hand, the ability to reason out contradictions without triviality characterizes LEI as a paraconsistent logic. Its properties can be studied in comparison with other members of this family. For instance, LEI is stronger than many other paraconsistent logics and admits a recursive semantics (in the sense of the meaning of an expression being determined by the meanings of its components). Such semantics, reflecting our intuitions on the notion of epistemic inconsistency, is given in the text. The calculus for LEI has been shown to be sound and complete with respect to it.

In the next sections the calculus for LEI will be presented followed by a discussion of the semantic intuitions assumed in its construction and by a precise formulation of its semantics.

2 THE CALCULUS OF EPISTEMIC INCONSISTENCY

The Calculus of Epistemic Inconsistency, CEI for short, is a paraconsistent calculus designed to cope with our intuitions about situations such as the ones described above. These intuitions are made precise in a semantics for the notion of *epistemic inconsistency* to be given later in this paper. Thus, this calculus is intended to reason out (meaningfully) the inconsistent theories arising on these situations. Its design aims to keep as many properties of classical logic as possible, without interfering with the properties required for the performance of this task.

Briefly stated, a paraconsistent logic is a logic in which triviality does not follow generally from contradiction. In its design CEI fulfills the requirements established in [Jaskowski 1948] and [da Costa 1974] concerning this kind of logical systems. As a matter of fact, the calculus CEI behaves classically for undoubting (monotonic, irrefutable) statements and paraconsistently for plausible (nonmonotonic, defeasible) ones. These two kinds of statements are distinguished in the language of the calculus by the use of an interrogation mark (?) suffixing the formulas of the last kind. When used in association with IDL, these marks are supplied by its default rules.

The following conventions are adopted in the presentation of CEI: the Greek letters α, β, γ denote arbitrary formulas in the language L' of the calculus, while the Roman capital letters A, B, C denote formulas in the language L without "?" (?-free formulas). $\neg \alpha$ will be used as a short for $\alpha \rightarrow p \wedge \neg p$, where p is an arbitrary sentential letter ("~" will be shown to behave as classical negation in CEI).

The axiomatics for CEI is the following:

- (i) $\alpha \rightarrow \beta \rightarrow \alpha;$
- (ii) $(\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma);$
- (iii)
$$\frac{\alpha \quad \alpha \rightarrow \beta}{\beta};$$
- (iv) $\alpha \wedge \beta \rightarrow \alpha;$
- (v) $\alpha \wedge \beta \rightarrow \beta;$
- (vi) $\alpha \rightarrow \beta \rightarrow \alpha \wedge \beta;$
- (vii) $\alpha \rightarrow \alpha \vee \beta;$
- (viii) $\beta \rightarrow \alpha \vee \beta;$
- (ix) $(\alpha \rightarrow \gamma) \rightarrow (\beta \rightarrow \gamma) \rightarrow (\alpha \vee \beta \rightarrow \gamma);$
- (x) $((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha;$
- (xi) $(\alpha \rightarrow B) \rightarrow (\alpha \rightarrow \neg B) \rightarrow \neg \alpha;$
- (xii) $\neg(\alpha \rightarrow \beta) \leftrightarrow \alpha \wedge \neg \beta;$
- (xiii) $\neg(\alpha \wedge \beta) \leftrightarrow \neg \alpha \vee \neg \beta;$
- (xiv) $\neg(\alpha \vee \beta) \leftrightarrow \neg \alpha \wedge \neg \beta;$
- (xv) $\neg \neg \alpha \leftrightarrow \alpha;$
- (xvi) $(\alpha? \rightarrow \beta?)? \rightarrow (\alpha? \rightarrow \beta?);$
- (xvii) $(\alpha \vee \beta)? \rightarrow \alpha? \vee \beta?;$
- (xviii) $(\neg \alpha)? \leftrightarrow \neg(\alpha?);$
- (xix) $\alpha \rightarrow \alpha?;$
- (xx) $\alpha?? \rightarrow \alpha?;$
- (xxi)
$$\frac{\alpha \rightarrow \beta}{\alpha? \rightarrow \beta?};$$
- (xxii)
$$\frac{\alpha}{\sim((\neg \alpha)?)}.$$

In the postulates above (iii), (xxi) and (xxii) are inference rules. The rule (iii) also holds as an implication, i. e., $\alpha \wedge (\alpha \rightarrow \beta) \rightarrow \beta$ is a theorem of CEI. The same is not true for the other two rules. That is why a single bar is used for (iii) and a doubled bar for the others. The same convention will be adopted for derived rules in CEI, in order to indicate whether the use of these rules affects eventual applications of the deduction theorem.

Special attention must be paid to the Roman letter B at axiom (xi). This is a key axiom for the attainment of the selective paraconsistency of CEI. It restricts the axiom of absurdity to undoubting formulas (formulas without "?").

For the construction of this calculus, some guidelines were assumed. For instance, the following schemes were definitely rejected:

$$\neg \alpha? \rightarrow \alpha \text{ (or even } \frac{\alpha?}{\alpha}),$$

$$\neg \alpha? \rightarrow (\neg \alpha)? \rightarrow \beta? \text{ (or } \frac{\alpha? \quad (\neg \alpha)?}{\beta?}).$$

By the first schema, defeasible knowledge would lead to irrefutable knowledge and, by the second, defeasible inconsistency would lead to (defeasible) trivialization. The implicative forms of the rules (xxi) and (xxii), for instance, would carry out $\alpha? \rightarrow \alpha$, while the rule $\alpha?, (\alpha \rightarrow \beta)? / \beta?$, although it may seem reasonable, entails $\alpha?, (\neg \alpha)? / \beta?$ in CEI. So, the distribution of "?" over the implication is not allowed. The same happens for $\alpha?, \beta? / (\alpha \wedge \beta)?$. In fact, these two rules can be shown to be equivalent in CEI.

The rule (xxi), used in conjunction with *modus ponens* rule, (iii), enables the derivation of $\beta?$ from $\alpha \rightarrow \beta$ and $\alpha?$. This kind of reasoning is one of the main mechanisms for propagation of "?" along inferences. Recall the example on birds and animals in the previous section. From the first default rule and the fact that Tweety is an animal, it follows " \neg fly(Tweety)?" (With IDL reasoning this would be defeated later but anyway the argument is illustrative on how rule (xxi) works.). The statement (2) in the example is instantiated as

$$\text{wing(Tweety)} \rightarrow \text{fly(Tweety)}$$

Taking the contrapositive and applying rule (xxi) we get

$$\neg \text{fly(Tweety)}? \rightarrow \neg \text{wing(Tweety)}?.$$

This allows the application of modus ponens to get " $\neg \text{wing(Tweety)}?$ ". The rule (xxii) states that we can't simultaneously be sure about α and consider that its classical negation is plausible, that is, all theory based on CEI having as theorems α and $(\neg \alpha)?$ is trivial. A particular case of this rule states that an irrefutable statement A should defeat its plausible negation $(\neg A)?$. This is done in IDL by the machinery of its default rules. Suppose that to the original statements of the Nixon's diamond example it is added that "anyone who promotes a

war is definitely not a pacifist", together with the information that Nixon promoted the Vietnam war. This would lead to a categorical " \neg pacifist(Nixon)", and it should defeat " $\text{pacifist}(Nixon)?$ ", otherwise the set of believes would become a trivial theory.

Let's give a look in the properties of CEI. Some theorems are in order. The first result states that the calculus CEI is in fact a classical calculus when restricted to undoubting formulas.

Theorem 1: All classical theorems hold to formulas of the language L (without "?") in CEI.

A corollary of this theorem is that the negation " \neg " behaves classically as regards formulas of L:

$$\begin{aligned} \vdash (A \rightarrow B) \rightarrow (A \rightarrow \neg B) \rightarrow \neg A; \\ \vdash \neg \neg A \rightarrow A. \end{aligned}$$

Theorem 2: The defined symbol " \sim " has indeed the properties of classical negation in CEI:

$$\begin{aligned} \vdash (\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \sim \beta) \rightarrow \sim \alpha; \\ \vdash \sim \sim \alpha \rightarrow \alpha. \end{aligned}$$

Theorem 3: All elimination and introduction rules for \wedge and \vee hold in CEI.

The following two theorems give an idea on CEI reasoning by showing samples of classical theorems which are kept and other which no longer hold in CEI, besides examples that illustrates its behavior.

Theorem 4: Among others, the following schemas are theorems or rules of CEI:

$$\begin{aligned} &\alpha \vee \neg \alpha; \\ &\neg \alpha \rightarrow \neg \alpha; \\ &(\alpha \rightarrow \beta) \leftrightarrow \neg \alpha \vee \beta; \\ &(\alpha \rightarrow \beta) \rightarrow \neg \alpha \vee \beta; \\ &(\alpha \vee \beta)? \leftrightarrow \alpha? \vee \beta?; \\ &(\neg \alpha)? \leftrightarrow \neg (\alpha?); \\ &(\alpha? \rightarrow \beta?) \rightarrow (\alpha \rightarrow \beta)?; \\ &(\alpha \rightarrow \beta?) \rightarrow (\alpha \rightarrow \beta)?; \\ &\alpha // (\alpha \rightarrow \beta)? \rightarrow \beta?; \\ &(\alpha \wedge \beta)? \rightarrow \alpha? \wedge \beta?; \\ &\alpha // \beta? \rightarrow (\alpha \wedge \beta)?; \\ &\neg (\alpha \wedge \neg \alpha). \end{aligned}$$

Notice that the thesis $\neg (\alpha \wedge \neg \alpha)$, which express the *non-contradiction law* (although in terms of a weak negation), is a theorem in CEI. Amazingly this does not prevent CEI of being paraconsistent, in spite of the condition of $\neg (\alpha \wedge \neg \alpha)$ not being a theorem has been stated in [da Costa 1974] as one requirement paraconsistent calculi should fulfill. As a matter of fact, it

is not a necessary requirement for the attainment of paraconsistency and, therefore, to avoid this theorem is against the aim of keeping as much classical theorems as possible. Furthermore CEI is not the only paraconsistent calculus in which $\neg(\alpha \wedge \neg\alpha)$ holds. This also happens to many relevant systems and to all the paraconsistent calculi in [Buchsbaum & Pequeno 1991].

Theorem 5: Among others, the following schemas are not theorems or rules of CEI:

- $(\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \neg\beta) \rightarrow \neg\alpha;$
- $\neg\alpha \rightarrow \neg\neg\alpha;$
- $\neg\alpha \vee \beta \rightarrow (\alpha \rightarrow \beta);$
- $\neg\alpha, \alpha \vee \beta / \beta;$
- $\alpha \rightarrow \beta / \neg\beta \rightarrow \neg\alpha;$
- $\alpha?, (\neg\alpha)? / \beta?;$
- $\alpha? / \alpha;$
- $\alpha?, (\alpha \rightarrow \beta)? / \beta?;$
- $(\alpha \rightarrow \beta)? / \alpha \rightarrow \beta?;$
- $\alpha? \wedge \beta? / (\alpha \wedge \beta)?;$
- $\beta? / \alpha \rightarrow (\alpha \wedge \beta)?.$

This last theorem is useful in showing arguments that are not allowed. For instance, suppose we have $A?$ and $(A \rightarrow B)?$. From the last we have $(\neg A \vee B)?$ and then $(\neg A)? \vee B?$, or even $\neg(A?) \vee B?$; but from this we cannot have $A? \rightarrow B?$, and then, by applying modus ponens, $B?$. On the other hand, by considering directly $A?$ and $\neg(A?) \vee B?$, also $B?$ does not follow.

Another important remark is that CEI reasoning does not mix up different extensions, as it should be expected. In a sense it is not adjunctive as regarding defeasible formulas. From $\alpha?$ and $\beta?$ we don't have $(\alpha \wedge \beta)?$, although having $\alpha? \wedge \beta?$. This property has a very significative effect on the reasoning. Suppose the following example:

- fly(x)? : fast(x)
fast(x)?
- \neg fly(x)? : cautious(x)
cautious(x)?
- fast(x) \wedge cautious(x) \rightarrow perfect-traveller(x)

From this last statement it can be derived (by rule xxi)

$$(fast(x) \wedge cautious(x))? \rightarrow \text{perfect-traveller}(x),$$

but it would not be fair from "fly(Tweety)?" and " \neg fly(Tweety)?" to conclude that Tweety is a perfect traveller and this is in fact not allowed in CEI. In other words, we have that may be Tweety is fast and may be Tweety is cautious, but not that may be Tweety is fast and cautious.

Every thing has a price to be paid for. The precautions against mixing up defeasible conclusions has a drawback, which ultimately comes from our decision of not distinguishing nor talking about extensions. There are many situations in which effectively we want to combine defeasible conclusions, that would be in a same extension.

For instance, suppose we get from a default that "has-hoof(Incitatus)?" and "quadruped(Incitatus)?", and that we also have

$$\text{has-hoof}(x) \wedge \text{quadruped}(x) \rightarrow \text{equine}(x)$$

Thus, we would like to be able to conclude "equine(Incitatus)?", but unfortunately this is not allowed for the feature just discussed.

A way out of this trouble would be to anticipate these valid combinations and, to remain safe, combine them through the default rules from which they come from. For instance, besides

$$\alpha : \text{has-hoof}(x) \quad \text{and} \quad \beta : \begin{cases} \text{quadruped}(x) \\ \text{has-hoof}(x)? \end{cases}$$

it would be added their combination:

$$\alpha \wedge \beta : \begin{cases} \text{has-hoof}(x) \wedge \text{quadruped}(x) \\ (\text{has-hoof}(x) \wedge \text{quadruped}(x))? \end{cases}$$

Now, "equine(Incitatus)?" can be inferred. Certainly this is not very elegant, but it is anyway effective in solving this problem.

Aiming at the construction of a theory of definition for CEI, a new abbreviation, the *strong implication*, is introduced: $\alpha \Rightarrow \beta$ is short for $(\alpha \rightarrow \beta) \wedge (\neg\beta \rightarrow \neg\alpha)$, as well as its corresponding double implication: $\alpha \Leftrightarrow \beta$ is short for $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

A replacement theorem for CEI, which would enable the eliminability of defined terms, can then be proved.

Theorem 6: (replacement) Let α' be a formula obtained from α by substituting β' for some occurrences (not necessarily all) of β . Then $\Gamma \vdash \beta \Leftrightarrow \beta'$ entails $\Gamma \vdash \alpha \Leftrightarrow \alpha'$.

The concept of a formula being closed is introduced now. It refers to a formula having all its components under the scope of an interrogation mark (taking " \neg " as in the language). A theorem assures that additional "?" suffixing this kind of formulas are irrelevant.

We say that a formula is *closed* if it has one of the forms $\alpha?$, $\neg\beta$, $\neg\neg\beta$ or $\beta \# \gamma$, where β and γ are closed formulas and $\# \in \{\rightarrow, \wedge, \vee\}$.

Theorem 7: If α is a closed formula, then $\vdash \alpha \Leftrightarrow \alpha?$.

This concept plays a role in a version of the deduction theorem which holds for CEI.

Theorem 8: (restricted deduction) If there is a deduction of β from Γ, α , then $\Gamma \vdash \alpha \rightarrow \beta$, unless α is not closed and the rules (xxi) or (xxii) are used after the first time α occurs justified by being a premise.

3 SEMANTICS

The basic intuition to be captured by the semantics of LEI is the truthfulness relative to multiple observations of a same phenomenon, taken under different conditions, when the informations about these conditions (or even on how the observations can be affected by them) are not available. It might happen, for instance, an experiment where a variation of conditions out of control (some variation is always the case in any experiment) is enough to affect the experiment to a level detectable by the instruments.

We are facing again a situation of insufficient knowledge leading to disagreement. It parallels our initial motivation about multiple extensions generated by a default theory when lack of knowledge does not enable the control of the selection among alternatives equally plausible.

A semantic framework reflecting the above scenario can be constructed as follows. A valuation V for an atomic proposition p is composed by a non-empty collection C of classical valuations for p (each one may be thought as expressing the opinion of an observer about p). There are two extreme alternatives for the definition of the truth value of p - a *credulous* and a *skeptical* one. By the credulous alternative p is taken as true if, for some classical valuation v belonging to C , $v(p)$ is true. This alternative leads to a *paraconsistent semantics*, which we call *semantics of maximization*. This is the kind of semantics adopted in LEI.

By the other alternative p would be true only if it is true for all v in C . This could lead to a *paracomplete semantics (semantics of minimization)*. A paracomplete logic is a logic in which the thesis expressing the excluded middle principle ($\alpha \vee \neg \alpha$) is not valid.

A combination of these two alternatives would lead to a *non-alethic semantics*. A non-alethic logic is a logic both paraconsistent and paracomplete. All these alternatives have been explored for the construction of semantical systems and their corresponding calculi in [Buchsbaum & Pequeno 1991].

The semantics for LEI is given by the definition of a valuation function V , in terms of auxiliary functions v_{\max} and v_{\min} , recursively defined for each classical valuation v belonging to C . Roughly speaking, this semantics corresponds to the adoption of the first alternative as regards defeasible formulas (formulas suffixed by "?") and the second alternative to undoubting (?-free) formulas. In other words, $V(A?)$ is true if A is true for some classical valuation v in C (A is true for someone), while $V(A)$ is true iff it is true for all classical valuations v in C (A is

true for everyone). The later intends to capture the idea that the set of (monotonic) theorems of Γ corresponds to the intersection of all possible extensions (completions) of Γ .

Definition: Let C be a non-empty collection of classical valuations. For each $v \in C$, let v_{\max} and v_{\min} be functions from L' to $\{0,1\}$ and let V be a function from L' to $\{0,1\}$ too, such as the following conditions are satisfied:

$$\begin{aligned} V(\alpha) &= 1 \text{ iff for all } v \in C, v_{\max}(\alpha) = 1; \\ v_{\max}(p) &= v_{\min}(p) = v(p); \\ v_{\max}(\neg \alpha) &= 1 \text{ iff } v_{\min}(\alpha) = 0; \\ v_{\min}(\neg \alpha) &= 1 \text{ iff } v_{\max}(\alpha) = 0; \\ v_{\max}(\alpha?) &= 1 \text{ iff for some } v' \in C, v'_{\max}(\alpha) = 1; \\ v_{\min}(\alpha?) &= 1 \text{ iff for all } v' \in C, v'_{\min}(\alpha) = 1; \\ v_{\max}(\alpha \rightarrow \beta) &= 1 \text{ iff } v_{\max}(\alpha) = 0 \text{ or } v_{\max}(\beta) = 1; \\ v_{\min}(\alpha \rightarrow \beta) &= 1 \text{ iff } v_{\max}(\alpha) = 0 \text{ or } v_{\min}(\beta) = 1; \\ v_{\max}(\alpha \wedge \beta) &= 1 \text{ iff } v_{\max}(\alpha) = 1 \text{ and } v_{\max}(\beta) = 1; \\ v_{\min}(\alpha \wedge \beta) &= 1 \text{ iff } v_{\min}(\alpha) = 1 \text{ and } v_{\min}(\beta) = 1; \\ v_{\max}(\alpha \vee \beta) &= 1 \text{ iff } v_{\max}(\alpha) = 1 \text{ or } v_{\max}(\beta) = 1; \\ v_{\min}(\alpha \vee \beta) &= 1 \text{ iff } v_{\min}(\alpha) = 1 \text{ or } v_{\min}(\beta) = 1. \end{aligned}$$

For any formula α in the language of LEI, the truth value of α is given by $V(\alpha)$.

Notice that, although the second alternative has apparently been adopted in our semantics for classical formulas, this does not lead to paracompleteness here, because V is defined in a way to make $\alpha \vee \neg \alpha$ valid.

The next is the key theorem in this section, stating the soundness and the completeness of the calculus with respect to the given semantics.

Theorem 9: (soundness and completeness)

$$\Gamma \vdash \alpha \text{ iff } \Gamma \Vdash \alpha$$

The following theorem assures that the interpretation for classical formulas (?-free formulas) given in LEI is in accordance with the classical propositional semantics. It states that, under reasonable conditions, the ?-free logical consequences of a set of formulas in LEI are the same as the classical logical consequences of the ?-free formulas in this set. The reasonable conditions refer to having in the set only ?-free formulas or formulas containing an "?" suffixing a ?-free formula (avoiding arbitrary mixing up of interrogation marks). These are the kind of formulas generated naturally by a default theory consisting of classical formulas and IDL default rules.

A collection of formulas of LEI is said *normal* if it is non trivial and all of its formulas have one of the forms A or A?

Theorem 10: Let Γ' be a normal collection of formulas in L' ($L + ?$) and Γ the set of $?$ -free formulas in Γ' . Then

$$\Gamma' \vdash_{LEI} A \text{ iff } \Gamma \vDash A,$$

where " \vDash " stands for the classical propositional logical consequence.

4 CONCLUSIONS

Our aim in this paper has been to demonstrate that the idea of assuming all the conclusions supplied by reasoning on incomplete knowledge, in spite of expressing inconsistent views, can be taken seriously. We are able to devise two possible objections to be raised against this idea. The first is a conceptual objection based on the argument that contradiction is indicative of error occurrence and therefore the efforts should be directed towards the correction of these errors and not in propagate them. It could be added that an inconsistent description is a description of nothing. The second is a technical objection based on the belief that, in any reasonable logic, inconsistent theories are trivial (everything is a theorem). We think that the logic presented here is evidence enough to remove the later objection. A logic able to perform reasoning in the presence of contradiction, without trivializing, being at the same time strong enough to make this reasoning useful, is perfectly possible. We would like to further discuss the first objection.

Contradiction is effectively a test for error. If a contradiction is deduced from a set of premises, this imply the inconsistency of this set. Semantically this means that no model can exist in which all of them are true. Thus it is a positive indication that these premises make a bad description of a given piece of a world. A similar practice is adopted in the methodology of the exact sciences. Contradictions among conclusions of a scientific theory, or among previsions of a theory and empirical observations, demand a revision of the hypothesis of the theory. To stay free of contradiction is one of the main methodological prescriptions of standard scientific practice.

The situation is quite diverse when common sense or artificial intelligence applications are considered. Then the inaccuracy of the knowledge is recognized in advance, and so the occurrence of contradictions does not provide such strong indication. It may demand an effort to get more precise informations, but this refinement cannot be done beyond the limits of the knowledge available at a given time. In spite of it, reasoning must be done and decisions taken.

This kind of situation reveals that the role of reasoning is not exactly to come up with conclusions to be assumed as true in situations satisfying the premises. This picture fits well deductive reasoning and it is so prevalent as a paradigm for thinking that it is often taken as a general expression for reasoning.

Actually, the role of reasoning is to perform an analysis of the epistemic relations within the knowledge. It is to compose and judge evidences, to resolve conflicts (when possible) and to come up with relations between evidences and possible conclusions. These conclusions, in opposition to deductive ones, can never be detached from the premises in support of them. As has been pointed out, situations that satisfy the premises may not satisfy the conclusions. So, this kind of conclusions, more than a statement about the state of affairs, is a statement about our knowledge on the state of affairs. A contradiction then means simply that there are evidences in support of A as well there are also evidences supporting $\neg A$. There is nothing so striking about it. After words it is not even a contradiction. What can be a little striking is to take these conclusions altogether as a set of believes. This is precisely what is done here, but with the precautions of distinguishing this kind of belief and of treating them with a suitable logic. That is why we make a point in insisting about marking defeasible conclusions.

Perhaps the main point in these objections could be summarized in the observation that taking decisions on basis of the assumption that contradictory statements are true sounds nonsense and might even be dangerous. Although being correct, the observation misses the point. First of all because this is more a statement about the pragmatics of the situation, about the use to be done with a piece of information, than an epistemic one. In the second place because, as it has been argued, defeasible conclusions are not really intended just to be assumed as true. We certainly agree that an inference of $A? \wedge \neg A?$ may add extra warning in the consideration of this kind of conclusion and this is reflected in the way the given calculus treats it.

It is worthwhile at this point to make a clear statement about the position we defend. We are not affirming here (at least not yet) that this is the only correct way to perform reasoning on such situations. The position we assume is a bit weaker. We just advocate, providing technical support, that this is a possible and legal way to do the things. On top of that, there is also a feeling that there are effectively situations in which this is really the appropriate approach to be taken. Another important point made here is the idea of providing a purely logical analysis of the reasoning, at least up to the point nonmonotonicity is itself accepted as logical. Our idea about default reasoning is to have it entirely performed by default rules extending a monotonic basis. LEI is offered here as a candidate for such a basis, but other alternatives could be the case. For instance, a paracomplete logic

could be adopted if the situation requires a more cautious disposition.

The exploration of this kind of alternative is one of the directions for future research that we devise. As a first step in this direction, the authors have developed a whole family of non classical calculi coping with variations on a common semantical framework. These calculi and a study of their properties are presented in [Buchsbaum & Pequeno 1991]. We are also working out first order extensions for LEI and for all those calculi. Another point to deserve attention is to provide a semantics for the whole logic, encompassing its nonmonotonic part.

Acknowledgements

The authors are thankful to Newton C. A. da Costa and Richard Sylvan (former R. Routley) for discussions and criticism on earlier versions of this paper, and to Armando Haeberer, André Buchsbaum and Gerson Zavenucha for their help with revision and editing.

References

- Arthur Buchsbaum & Tarcisio Pequeno (1991) *Uma Família de Lógicas Paraconsistentes e/ou Paracompletas com Semânticas Recursivas*. To be published.
- Newton C. A. da Costa (1974) *On the Theory of Inconsistent Formal Systems*. Notre Dame Journal of Formal Logic, no. 15.
- S. Hanks & D. McDermott (1987) *Nonmonotonic Logic and Temporal Projection*. Artificial Intelligence, no. 33.
- S. Jaskowski (1948) *Un Calcul des Propositions pour les Systèmes Déductifs Contradictoires*. Studia Soc. Scien. Torunensis, section A, 1.
- P. H. Morris (1988) *The Anomalous Extension Problem in Default Reasoning*. Artificial Intelligence, no. 35.
- Tarcisio Pequeno (1990) *A Logic for Inconsistent Nonmonotonic Reasoning*. Technical Report 90/6, Department of Computing, Imperial College, London.
- Raymond Reiter (1980) *A Logic for Default Reasoning*. Artificial Intelligence, no. 13.
- N. Rescher & R. Brandon (1980) *The Logic of Inconsistency*. Blackwell, Oxford.

THE UTILITY OF CONSISTENCY-BASED DIAGNOSTIC TECHNIQUES

Gregory M. Provan

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389
USA

David Poole*

Department of Computer Science
University of British Columbia
Vancouver, BC
Canada V6T 1W5

Abstract

Within diagnostic reasoning a number of definitions of diagnosis such as minimal diagnosis, kernel diagnosis, and optimal (or most likely) diagnosis have been proposed. Most of these approaches assume that diagnoses *must* be computed, and that what should be computed can be defined *a priori*, independent of the use of the diagnosis and of the domain (i.e. from "first principles"). It was recently argued [Poole and Provan, 1990] that the diagnostic problem, as currently posed, is incomplete: it does not consider how the diagnosis is to be used, tests which can be done to clarify the hypothesised diagnoses, or the utility associated with the treatment of the abnormalities. This paper shows that purely logic-based definitions of diagnosis can define only the space of possible hypotheses consistent with (or entailed by) the evidence; computing minimal, kernel or most likely diagnoses requires using some extra-logical preference criterion; hence, these "first principles" definitions of diagnosis are not as general as has been claimed, but are problem-specific. A more general framework, that of partitioning the space of possible diagnoses into equivalence classes of diagnoses, is proposed. Consistency-based diagnoses, shown to be one class of diagnoses, are contrasted with another class, use-equivalent diagnoses.

1 INTRODUCTION

Diagnostic reasoning is an important application for AI research. Recently there have been a number of formalisations of the notion of diagnosis. These typically define the diagnostic task as computing the "logical possibilities" or the most likely of these. Within this

framework, there have been probabilistic definitions of diagnosis [Pearl, 1987; Peng and Reggia, 1987b; Peng and Reggia, 1987a], as well as abductive definitions of diagnosis [Poole, 1989a; Reggia *et al.*, 1985], minimal diagnosis [Reiter, 1987; de Kleer and Williams, 1987; de Kleer *et al.*, 1990], and kernel diagnosis [de Kleer *et al.*, 1990]. Unlike earlier logic-based diagnoses that consider what can be proven about a faulty device [Genesereth, 1984], these papers have considered that the question "what is a diagnosis?" is important to answer. The intuition is that it is important to characterise the set of "logical possibilities" for a diagnosis, presumably to be able to compare them. Most of these approaches assume that the diagnoses (or perhaps the most likely diagnoses) *must* be computed, and that what should be computed can be defined *a priori*, independent of the use of the diagnosis and of the domain, based only on "first principles" [Reiter, 1987].

It was recently argued [Poole and Provan, 1990] that the diagnostic problem, as often currently posed, is incomplete: it does not consider (1) how the diagnosis is to be used, (2) tests which can be done to clarify the hypothesised diagnoses, or (3) the utility associated with the treatment of the abnormalities. The fact that diagnostic reasoning is a *purposive* activity, in which the reasons for computing the diagnosis affect what is computed, has been ignored.

However, even if diagnostic reasoning is considered purely within the limited scope of finding the logical possibilities, serious problems arise in proposed definitions of a diagnosis. Poole and Provan [1990] studied six approaches to diagnostic reasoning, as well as their associated notions of optimality (most of which are based on probability theory). Given a problem, each approach defines a space of possible diagnoses, from which one can select a subset as optimal based on some notion of optimality. The six approaches examined were: (1) most likely single-fault hypothesis; (2) most likely posterior hypothesis [Pearl, 1986]; (3) most likely interpretation [Pearl, 1987; de Kleer and Williams, 1987; Peng and Reggia, 1987b; Peng and

*This author was supported by NSERC grant OGPOO44121.

Reggia, 1987a]; (4) probability of provability [Reiter, 1987; de Kleer *et al.*, 1990]; (5) covering explanations [Lin and Goebel, 1990]; and (6) utility-based explanation [Heckerman and Horvitz, 1990].

In analysing the diagnoses computed by these proposals, the different definitions of optimal diagnosis can be shown to have different qualitative (and quantitative) results, even given the same input data. If these definitions of diagnosis are indeed from “first principles” and are use-independent, then, presumably, they should compute the same conjunction of hypotheses as “a diagnosis”, given the same input data.

In particular, the logic-based definitions of diagnosis (approaches (4) and (5) above) claim to remain agnostic with respect to the minimal or kernel diagnoses computed; i.e. no criteria other than logical entailment are used to compute the diagnoses. Hence these definitions should be insensitive to how the diagnoses are to be used.

This paper shows that purely logic-based definitions of diagnosis can define only the space of possible hypotheses consistent with (or entailed by) the evidence; computing minimal, kernel or most likely diagnoses requires using some extra-logical preference criterion which explicitly or implicitly makes a commitment to the use of the diagnosis. We argue that such definitions are not agnostic, and that there is no use-free definition of minimal, kernel or most likely diagnosis. Hence, these “first principles” definitions of diagnosis are indeed not as general as has been claimed, but are problem-specific. It is argued that it is an inherent preference criterion which causes the different definitions of diagnosis to produce different answers to a given problem.

This paper argues that a utility-based approach is essential for diagnostic reasoning. Making the utility function explicit clarifies the entire diagnostic process, and shows the assumptions underlying many approaches to diagnostic reasoning which *implicitly* use utility functions. The particular focus of this paper is identification of the uses, and inherent utility functions, of well-known definitions of consistency-based diagnosis.

This paper also introduces a new characterisation of how the space of possible diagnoses can be partitioned into equivalent sets of diagnoses. It is shown that consistency-based diagnosis represents one of many partitioning methods. A partitioning based on use-equivalence of diagnoses is introduced. Two diagnoses are use-equivalent if the action taken (for example, treatment or component replacement) is the same for both diagnoses. These distinctions between use-diagnoses and the traditional consistency-based diagnoses are clarified when these classes are compared and contrasted.

The remainder of the paper is organised as follows. Section 2 introduces the notation, explores the possible uses of a diagnosis, and defines consistency-based and utility-based diagnoses. Section 3 analyses two formalisations of consistency-based diagnosis, as formalised in [Reiter, 1987; de Kleer *et al.*, 1990], and in [de Kleer and Williams, 1987]. Section 4 introduces a more general characterisation of diagnoses, which partitions the diagnoses into equivalence classes. Section 5 summarises the conclusions.

2 WHAT IS THE DIAGNOSTIC PROBLEM?

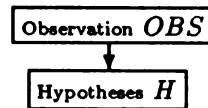
2.1 Notation

Let E be the knowledge used to compute a diagnosis. E can be broken down into a set SD of facts which are unchanging from instance to instance¹ (e.g. SD can be a model of a circuit which is being diagnosed), and a set OBS of observations concerning a particular instance. Let $H = \{h_1, \dots, h_m\}$ be the set of hypotheses under consideration given E . $T = \{t_1, \dots, t_l\}$ is the set of possible treatments.

2.2 Diagnostic Problem

Figure 1 depicts the current notion of diagnostic reasoning as formalised in the literature.

Figure 1: Existing formalisation of the diagnostic process

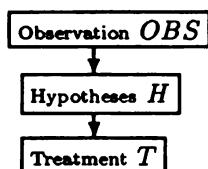


A more complete diagnostic process consists of reasoning from observations OBS (the evidence) to hypothesised diagnosis H , and then administering a treatment $\tau \in T$ for the diagnosis (or abnormalities). Tests can be performed to provide more observations. A depiction of this diagnostic process is shown in Figure 2.

In most current formal theories of diagnosis (e.g. [Reiter, 1987; Peng and Reggia, 1987b; de Kleer and

¹The term SD corresponds to the system description SD of [Reiter, 1987; de Kleer *et al.*, 1990], and is used to maintain consistency with those papers. However, SD is a more general description, as it is also appropriate when, for example, we have cause-symptom knowledge that hardly warrants the term “system description”.

Figure 2: A more complete diagnostic process



Williams, 1987; Pearl, 1987; Pople, 1982]), the treatment phase is not considered at all, and diagnoses are defined with respect to the evidence-hypothesis phase only. These approaches ignore utility considerations totally.²

We argue that diagnostic reasoning must take into account the complete cycle, and thus must be a utility-maximisation approach. In this approach, the definition of a diagnosis is strongly influenced by how the diagnosis is to be used.

The variety of possible uses of a diagnosis expands the notion of treatment to include:

1. finding out the thing (or things) that is (are) wrong; that is, through testing, to determine the exact state of the world with respect to the symptoms observed;
2. giving a plausible account (an explanation) for the symptoms; that is, to give a description of what is wrong that is understandable to people;
3. enabling a decision as to how to fix something; that is, to maximise the utility derived from the diagnostic process through the treatment of the abnormalities;
4. to fix the symptoms; in some cases we may be happy to fix the symptoms without caring about what is really wrong.

While each of these may seem reasonable, we will see that different goals will lead to different evaluation criteria. For example, the best decision may involve averaging over many cases, which may not be conducive to a good explanation. Also there may be no point in finding the exact causes for a problem if further refinement of the problem will not help either treatment or explanation.

It is important to keep these different possible goals in mind when considering different proposals. Each of these uses entails a particular utility function. Making the utility function explicit clarifies the entire diagnostic process, and shows the assumptions underlying each approach.

²It is interesting to note that in one of the earliest analyses of medical diagnostic reasoning, Ledley and Lusted [1959] described a method of implementing a diagnostic cycle including treatments.

2.3 Example: Structure of Medical Diagnosis

Everyday medical diagnosis demonstrates clearly the non-sequential nature of diagnostic reasoning: the system under study, the human body, is a homoeostatic system, one with feedback that tends to a natural equilibrium given inputs to the system. The response of such complex systems to treatment is thus not entirely predictable, as the treatment of one symptom may lead to the system itself altering other symptoms, sometimes adversely.

Medical diagnosis, best framed as an *hypothetico-deductive process* [Elstein *et al.*, 1978], involves the following tasks [Shortliffe and Perrault, 1990]:

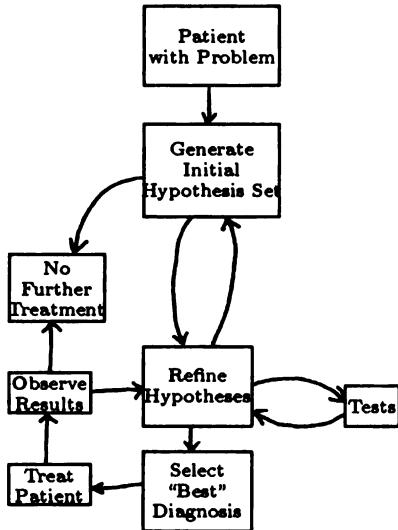
The central idea is one of sequential, staged data collection, followed by data interpretation and the generation of hypotheses, leading to hypothesis-directed selection of the next most appropriate data to be collected. As data are collected at each stage, they are added to the growing database of observations and are used to reformulate or refine the active hypotheses. This process is iterated until one hypothesis reaches a threshold level of certainty (for example, it is proven to be true, or at least the uncertainty is reduced to a satisfactory level). At that point, a management, disposition, or therapeutic decision can be made.

Figure 3 is a detailed representation of this sequential process. Note that the first stage of diagnosis consists of formulating an initial hypothesis set. This is followed by a process of hypothesis refinement, after which an optimal (e.g. most likely) diagnosis is selected. The results of administering a treatment based on this diagnosis are monitored, so that re-diagnosis and/or re-treatment may be done as required. Hence, both the test results and the observed results of the treatment are used for hypothesis refinement. A simpler representation of these stages is shown in Figure 3.

For example, consider the following situation in which a reasonable, initial hypothesised diagnosis is found to be incorrect following subsequent testing. A 45 year-old, obese man comes into an emergency room with shortness of breath and adrenalin-caused sweating (not due to high environmental temperatures). The initial diagnosis is that a heart attack is likely, and preparations are made to deal with that possibility.

However, further tests subsequently disconfirm that initial hypothesis. The patient is found to have blood in his urine and a high haemoglobin count. A second diagnosis is hypothesised, that of a bleeding ulcer. Significant blood loss from such a condition would cause shortness of breath and sweating. Further tests confirm this second hypothesis.

Figure 3: A complete medical diagnostic cycle as proposed by [Shortliffe and Perrault, 1990]



This example shows that an initial best hypothesis may be completely wrong, and that a mixed strategy, in which further tests and treatments serve to confirm/disconfirm the initial hypothesis, is necessary.

3 PREFERENCE CRITERIA INHERENT IN MINIMAL AND KERNEL DIAGNOSES

This section shows that consistency-based definitions of diagnoses, such as minimal and kernel diagnoses, implicitly impose a problem-specific utility function. Hence, the consistency-based definition of the space of logical possibilities is shown not to be a general, but a problem-specific, definition. The most precise definitions, those of Reiter ([1987],[de Kleer *et al.*, 1990]), will be analysed.

3.1 Reiter's Consistency-Based Diagnosis

3.1.1 Definitions

Reiter [1987] proposed a consistency-based definition of diagnosis as a means of computing a diagnosis “from first principles”. This proposal has subsequently been clarified and extended in [de Kleer *et al.*, 1990]. The analysis of this approach is taken primarily from [de Kleer *et al.*, 1990].

Similar to other consistency-based approaches, Reiter’s approach computes a set of “minimal” diagnoses which are logically consistent with $SD \cup OBS$, a set

of observations (OBS) and a system description (SD). The notion of minimality and the preference criteria used to determine minimality will be precisely defined shortly.³

Reiter defines a system to be a pair $(SD, COMPS)$, where SD is the system description, a set of first order sentences, and $COMPS$ is the set of system components, a finite set of constants. An observation OBS is a finite set of first order sentences, so that the system and observations are given by $(SD, COMPS, OBS)$. $AB(c)$ ⁴ is the proposition that component c is working abnormally (see [Poole, 1989b; de Kleer *et al.*, 1990]). A clause is a disjunction of literals with no pair of complementary literals (e.g. x and \bar{x}). An AB -clause is a disjunction of AB -literals with no pair of complementary AB -literals.

In this terminology, we have

Definition 3.1 A *diagnosis* for $(SD, COMPS, OBS)$ is a minimal set $\Delta \subseteq COMPS$ such that

$$\begin{aligned} SD \cup OBS \cup \{AB(c) \mid c \in \Delta\} \\ \cup \{\neg AB(c) \mid c \in COMPS - \Delta\} \end{aligned}$$

is consistent.

In other words, a diagnosis consists of a set of malfunctioning components such that the assumption of this set of abnormal components together with the assumption of normal functioning of the other components is consistent with the system description and observation.

Definition 3.2 A *minimal diagnosis* is a diagnosis without any proper subsets that are also diagnoses.

A kernel diagnosis is subsequently defined in [de Kleer *et al.*, 1990] using definitions of conflicts and minimal conflicts:

Definition 3.3 A *conflict* of $(SD, COMPS, OBS)$ is an AB -clause entailed by $(SD \cup OBS)$, i.e. a disjunction of AB literals with no literal repeated.

Definition 3.4

A *minimal conflict* M of $(SD, COMPS, OBS)$ is a conflict no proper subclause of which is a conflict of $(SD, COMPS, OBS)$

A kernel diagnosis is defined in the following section.

³Note that in the terms of [Poole and Provan, 1990], this approach does not compute an *optimal* diagnosis, but enumerates the *set of minimal* diagnoses. Hence, this approach cannot compute a single optimal diagnosis unless there exists a single minimal diagnosis.

⁴The AB predicate is due to McCarthy [McCarthy, 1986].

3.1.2 Prime Implicant Characterisation

An important contribution of [Reiter, 1987; de Kleer et al., 1990] is the characterisation of diagnoses, minimal diagnoses, etc. in terms of the mathematically precise and well-understood notions of implicants and prime implicants. This section reviews these notions.

In the propositional case (on which we will focus in the rest of the paper), an implicant γ of a set Σ of clauses is a conjunction of literals containing no pair of complementary literals such that $\Sigma \models \gamma$. A prime implicant ψ of Σ is an implicant with no strict subset which is also an implicant with respect to Σ . Ψ is the set of prime implicants of Σ .

Using these definitions, we have:

Definition 3.5 A diagnosis is the set Γ of implicants of $SD \cup OBS$.

Definition 3.6 \mathcal{D} is a minimal diagnosis of $(SD, COMPS, OBS)$ iff $\bigwedge_{c \in \Delta} AB(c)$ is a prime implicant of the set of positive minimal conflicts of $(SD, COMPS, OBS)$ for $\Delta \subseteq COMPS$.

Definition 3.7 A kernel diagnosis is given by:

$$\kappa(SD, COMPS, OBS) = \Psi(M(SD \cup OBS)).$$

If the minimal conflicts are all positive, then the minimal diagnosis corresponds to the kernel diagnosis [de Kleer *et al.*, 1990].

The key observation about all of these definitions of diagnosis is that they are based on prime implicants. Consequently, in the following analysis, the definition of preference criterion will be that inherent in the computation of Ψ .

3.2 Consistency-Based Diagnosis Preference Criteria

It is noted in [de Kleer *et al.*, 1990, p.329] that the “analysis ... corresponds exactly to the classical analysis...of minimization of Boolean functions”. Given a Boolean expression F (e.g. composed of the conjunction of a set Σ of clauses), Boolean minimisation entails computing a minimal Boolean expression \mathcal{F} equivalent to F . Minimality is determined with respect to a cost function (or criterion) C , and is done in two stages [McCluskey, 1956]:

1. The implicants $\Gamma(\Sigma)$ are computed, and from these the prime implicants $\Psi(\Sigma)$ are determined.
 2. From $\Psi(\Sigma)$ a set $\Psi^*(\Sigma)$ of prime implicants are selected, according to the cost criterion C , to compose the minimal expression. In other words, some set $\Psi^*(\Sigma) \subseteq \Psi(\Sigma)$ is selected such that $F = \bigvee_{\psi_j \in \Psi^*} \pi_j$, and F is the least cost expression equivalent to E .

Both $\Psi(\Sigma)$ and $\Gamma(\Sigma)$ are logically equivalent to Σ . Note that the implicants are simply clauses entailed by Σ , i.e. they are consistency-based and *use-independent*. The selection of prime implicants, however, loses the use-independence, since a particular type of implicant is selected, according to \mathcal{C} . It can be shown that \mathcal{C} cannot in general be defined to identify a use-independent minimal diagnosis. Common criteria include:

1. C_1 : Minimum number of appearances of literals in the expression, i.e. the clauses which minimise the total number of literals in $\Psi(\Sigma)$.
 2. C_2 : Minimum number of literals appearing in each clause.
 3. C_3 : Minimum number of clauses in an expression (provided there is no other such expression with the same number of terms and with fewer literals).
 4. C_4 : Minimum number of unnegated literals in an expression.

The third criterion is adopted in most descriptions of this process (e.g. Kohavi [1978]), and is also adopted in [de Kleer *et al.*, 1990]. However, no mention is ever made of the underlying cost criterion in [de Kleer *et al.*, 1990]. This section shows that the cost criterion chosen can indeed make a difference. That is, if Ψ_{C_i} is the set of prime implicants chosen under criterion C_i , $\Psi_{C_k} \neq \Psi_{C_l}$, $k \neq l$.

The same preference criterion is used for selecting both minimal and kernel diagnoses in [de Kleer *et al.*, 1990]. In general there are multiple sets of kernel diagnoses, and the minimal diagnoses are a subset of the kernel diagnoses. In the case where all minimal conflicts are positive, there is one set of kernel diagnoses, namely the minimal set of diagnoses.

Example

In these examples, for notational convenience a literal x_i means $AB(x_i)$, and conjunctions of literals are denoted by their product, e.g. x_1x_2 .

Example 3.8 Consider the following set of prime implicants over a set of variables $\{x_1, x_2, x_3, x_4\}$.⁵

$$\Psi = \{\overline{x_2x_3}, \overline{x_2x_4}, x_3\overline{x_4}, x_2x_4\}.$$

Using Criterion C_3 produces two kernel diagnoses:

$$\begin{array}{lll} K_1 & = & \overline{x_2x_4} \quad \vee \quad x_2x_4 \quad \vee \quad \overline{x_2x_3} \\ K_2 & = & \overline{x_2x_4} \quad \vee \quad x_2x_4 \quad \vee \quad \overline{x_3x_4} \end{array}$$

However, using a criterion C^+ (which selects a minimal expression containing the smallest number of unnegated AB-literals) will produce just the first of the

⁵The DNF formula producing this set of prime implicants is $F = \overline{x_1x_2x_3x_4} \vee \overline{x_1x_2\overline{x}_3x_4} \vee \overline{x_1\overline{x}_2x_3\overline{x}_4} \vee \overline{x_1\overline{x}_2x_3x_4} \vee \overline{x_1x_2\overline{x}_3x_4} \vee \overline{x_1x_2x_3\overline{x}_4} \vee x_1\overline{x}_2\overline{x}_3\overline{x}_4 \vee x_1x_2\overline{x}_3x_4 \vee x_1x_2x_3\overline{x}_4$.

kernel diagnoses (K_1), since under this criterion the second kernel diagnosis is *not* minimal. This is because K_1 has 2 unnegated literals, and K_2 has 3 unnegated literals; K_1 and K_2 differ in the third conjunction.

Note that under C^+ minimal diagnoses would always remain the same as under C_3 , since minimal diagnoses contain unnegated AB-literals only. In some sense, C^+ is a more reasonable criterion than C_3 , as one is really interested in the abnormal components (unnegated AB-literals), and not in both the normal and abnormal components; only the abnormal components need to be fixed. C_3 minimises with respect to the sum of the normal and abnormal components.

Example 3.9 Consider the following set of prime implicants over a set of variables $\{x_1, x_2, x_3, x_4\}$:

$$\Psi = \{\overline{x_1}x_2\overline{x_3}, \overline{x_1}x_2x_4, \overline{x_2}x_3, x_3x_4, x_1x_3\}.$$

Using criterion C_3 produces a unique kernel diagnosis:

$$\overline{x_1}x_2\overline{x_3} \vee \overline{x_2}x_3 \vee x_1x_3 \vee x_3x_4.$$

However, criterion C^+ produces two minimal diagnoses:

$$\begin{aligned} &\overline{x_1}x_2\overline{x_3} \vee \overline{x_2}x_3 \vee x_1x_3 \vee x_3x_4 \\ &\overline{x_1}x_2\overline{x_3} \vee \overline{x_2}x_3 \vee x_1x_3 \vee \overline{x_1}x_2x_4. \end{aligned}$$

The kernel diagnosis set does not even contain a minimal diagnosis computed under C^+ . This shows that a relatively small change in the preference criterion can cause quite different sets of diagnoses to be computed.

Analogous examples could be presented contrasting the minimal diagnoses computed under criteria C_1 and C_2 with those computed under C_3 (i.e. kernel diagnoses).

3.3 Challenges to Consistency-based Diagnosis

It is a reasonable question to ask what assumptions must be made for this notion of diagnosis to be appropriate. This then provides information about the domains to which this theory is applicable.

Sequential diagnosis The computational complexity of incorporating new information using this approach limits its usefulness for sequential diagnosis. Sequential diagnosis entails incorporating new evidence into the existing problem description, which in this formalisation requires recomputing sets of prime implicants. Given a new clause Σ' added to a clause set Σ (assuming that the set of prime implicants for Σ , $\Psi(\Sigma)$, has already been computed) recomputing the prime implicants for $(\Sigma' \cup \Sigma)$ using $\Psi(\Sigma)$ is as computationally expensive (in the worst case) as computing $\Psi(\Sigma' \cup \Sigma)$ from scratch [Kean and Tsiknis, 1990]. There may be an exponential number of

implicants, and computing $\Psi(\Sigma)$ is NP-hard even if there is a sub-exponential number of implicants [Provan, 1990]. Hence repeating such a computation is bound to be a computationally expensive task. However, it is possible that approximation algorithms may provide efficient computation methods, at the expense of guaranteed solutions. As such, this approach is limited to domains in which sequential diagnosis is not essential, excluding applications like medicine, where testing and sequential diagnosis plays an important role.

Hierarchical diagnosis Since the approach “flattens out” the hypothesis space, there is no ability to enable hierarchical diagnosis, so achieving reduction of complexity.

Reasonable diagnoses This approach assumes that the minimal or kernel diagnoses provide a definition of reasonable diagnosis, i.e. a diagnosis which optimises the underlying utility function. In general, there is no notion of utility function to determine which of the diagnoses maximises that function. Such a utility function may indicate the optimal diagnosis, or that no diagnosis is adequate. For example, if the value of the utility function is sufficiently low, then that means that the diagnosis is not good enough to define a treatment, and tests must be done to gather new information. It may be necessary to use a threshold to select the set of diagnoses which provide an accurate measure of reasonable explanations for the evidence [Peng and Reggia, 1987c].

3.4 Consistency-Based Diagnosis with Sequential Testing

de Kleer and Williams have proposed a theory of diagnosis [de Kleer and Williams, 1987], which has been implemented within an ATMS-based system called the General Diagnostic Engine (GDE). This theory is referred to as the General Diagnostic Theory (GDT). GDT uses a logical formalisation of a system as does Reiter's theory, and makes assumptions about the correct functioning of the components of the system (e.g. A_{M_1} is the assumption that component M_1 is functioning). Each assumption is the dual to an AB-literal, e.g. A_{M_1} is the dual of $AB(M_1)$. The aim of the program is to find the most likely assignment of truth values to every hypothesis (i.e., the most likely diagnoses). This is what Pearl [1987], Peng and Reggia [1987b] and Cheeseman [1990] compute, and is different to the most likely minimal or kernel diagnosis (see [Poole and Provan, 1990] for a discussion of the differences). de Kleer and Williams use tests in a sequential manner, but do not consider the use of the diagnosis. They assume that they want the “maximum information” tests. This is different to performing a test that has the maximum payoff (taking utilities into account). GDT may conduct tests to discriminate between diagnoses

that would have the same treatment. This would be a wasted test.

4 AN ANALYSIS OF DIAGNOSTIC EQUIVALENCE CLASSES

This section formalises the notion a diagnostic equivalence classes, and of a use-equivalent class, i.e. a class in which each member has the same treatment. In this section we use the normal definition [Chang and Lee, 1973] of an interpretation (an assignment of truth values to the propositions) and a model of a set of formulae (an interpretation in which all of the formulae are true). The idea used is the basis of modern logic, namely that the models of a set ϕ of formulae are the possible ways that the world could be, given that all we know is that ϕ is true. This idea is exploited in investigating consistency-based diagnoses.

4.1 Consistency-based diagnosis

In this section, of a class of consistency-based diagnoses is defined. This provides a method for characterising consistency-based diagnoses, and distinguishing consistency-based diagnoses from other definitions of diagnosis.

Definition 4.1 A diagnosis space, \mathcal{D} , is a set of formulae. An element of a diagnosis space, $D \in \mathcal{D}$, is called a possible diagnosis.

Different consistency-based approaches to diagnosis are characterised by different diagnosis spaces. The main idea is that the diagnosis space is the space of possible diagnoses that is pruned by (in)consistency.

A problem is characterised by a formula SD representing the system description, and a formula OBS representing the observations of an instance of the system. The diagnosis space can be a function of the system, but is not a function of OBS .

Definition 4.2 A consistency-based diagnosis \mathcal{D}_{CB} , is an element of the diagnosis space that is consistent with $SD \wedge OBS$. That is, if \mathcal{D} represents the diagnosis space, the set of consistency-based diagnoses is given by

$$\mathcal{D}_{CB} = \{D \in \mathcal{D} : SD \wedge OBS \wedge D \text{ is consistent}\}.$$

Note that there are other forms of diagnosis that are not in the class of consistency-based diagnoses, for example

- abductive diagnosis [Poole, 1988; Poole, 1989b], where the space of diagnoses is a function of the observations. In this approach the possible diagnoses are generated in order to explain the observations; hence, the space of possible diagnoses depends on the observations.

- Bayesian analyses of diagnosis (e.g., [Pearl, 1988, §5.4-5.7]) where the notion of “a diagnosis” is not used. In essence, this approach averages over all of the possible states [Cheeseman, 1990], rather than giving something the status of a diagnosis (although this can be done, as shown in section 4.6).

In the best developed theory of consistency-based diagnosis [de Kleer *et al.*, 1990], the diagnosis space is the set of assignments of truth values to the propositions $AB(C)$ where C is a “component”. As stated in [de Kleer *et al.*, 1990, p. 329], the “overall objective is to find methods of characterising all diagnoses”. The next section analyses the ways in which this definition of a diagnosis may be appropriate, and points out that the characterisation of the space in terms of kernel diagnoses hides some distinctions that make diagnoses useful.

Before beginning the analysis, let us consider useful properties that the space of all “diagnoses” should have.

1. The sub-space consistent with a system (i.e., the set of diagnoses) should convey all of the information about the state of the system that is necessary for subsequent use (treatment and/or description of the problems). There seems to be little point in defining the notion of an intermediate stage called a “diagnosis” if we have to keep going back to refer to the system itself.
2. It should provide a space over which we can search by performing tests. Tests can potentially be used to divide the space, pruning some diagnoses until we arrive at the “correct” diagnosis, or until we decide that it is not worthwhile pruning the space any more.
3. It can perhaps even be used to provide a space over which we can apply a measure to determine the probabilities of diagnoses.

4.2 Loss of information

It is important to notice that information is lost in computing the set of all diagnoses. Whereas all models of $SD \cup OBS$ are models of some diagnosis ($SD \cup OBS$ implies the disjunct of the diagnoses), the opposite is not true; not all models of a diagnosis are models of $SD \cup OBS$ (the disjunct of the diagnoses does not imply $SD \cup OBS$). An implicit assumption in the consistency-based definition of diagnosis is that the lost information is not useful.

Even though the definition of \mathcal{D}_{CB} in [de Kleer *et al.*, 1990] lost information going from the system to the “diagnoses”, great care was taken to ensure no information loss in going from the sets of diagnoses to the

descriptions of the space of all diagnoses (the minimal, kernel and prime diagnoses). The problem with the minimal and prime diagnoses is that the descriptions lose information which in fact is important, and not inconsequential (as assumed implicitly in [de Kleer *et al.*, 1990]).

Any formula defined as “a diagnosis” corresponds to a set of possible states of the system (i.e., the models of the diagnosis intersect with a set of the models of the system). A set of interpretations is grouped into one class and that set of interpretations is given an identity as “a diagnosis”. The sets of models given an identity by claiming they form an x -diagnosis (where x is “minimal”, “kernel”, etc.) are now analysed.

4.3 Partial and Kernel Diagnoses

The partial diagnoses of [de Kleer *et al.*, 1990] are the descriptions with the property that any model of the description is a model of $SD \cup OBS$. Computing the set of minimal diagnoses from the set of diagnoses entails combining descriptions of the form

$$\begin{aligned} & A \wedge B \\ & A \wedge \neg B \end{aligned}$$

into one class that is labelled with A .

The kernel diagnoses [de Kleer *et al.*, 1990] are the minimal partial diagnoses. The set of kernel diagnoses covers the same space of possibilities as the set of all diagnoses. It therefore seems reasonable to use the kernel diagnoses to define the space of all diagnoses. Unfortunately, as shown below, this does not seem to help carry out the activity of diagnosis.

Consider the following example:

Example 4.3 Suppose we have a system and observation that have the following diagnoses:

$$\begin{aligned} D_1 : & AB(A) \wedge AB(B) \wedge AB(C) \\ D_2 : & AB(A) \wedge AB(B) \wedge \neg AB(C) \\ D_3 : & AB(A) \wedge \neg AB(B) \wedge AB(C) \end{aligned}$$

The two kernel diagnoses are given by the following table:

Kernel Diagnoses	Diagnosis Set
$AB(A) \wedge \neg AB(B)$	$\{D_1, D_2\}$
$AB(A) \wedge AB(C)$	$\{D_1, D_3\}$

There are some important points to notice from this example:

1. The kernel diagnoses are covering, but not disjoint. A problem can arise in trying to refine the diagnoses to determine the correct diagnosis. It could be the case that a number of diagnoses

are all correct. As a result, this notion of a kernel diagnosis is incorrect if a “differential diagnosis” (e.g. through conducting further tests) is required.

2. With the original definition of diagnoses we can determine the probability of the truth of an hypothesis by summing the probabilities of the diagnoses in which that hypothesis is true. That is for any formula Q that uses only the AB predicate,

$$p(Q) = \sum_{\substack{D: D \text{ is a diagnosis}, \\ D \models Q}} p(D).$$

This option is lost when we have only the kernel diagnoses. One reason is that the kernel diagnoses are not disjoint, and so we cannot just sum the probabilities. The second reason is that we cannot determine the truth of a particular hypothesis, as the kernel diagnosis may say nothing about the truth of the hypothesis.

3. A dangerous condition may arise in this example when $AB(B) \wedge AB(C)$ is true. As this condition is covered by all the diagnoses, the diagnoses say nothing about this condition (which may be the main thing to be identified). Because the notion of Kernel diagnosis is purely syntactic, it does not allow “tuning” of the diagnoses to focus on what is important for the diagnostician. In fact, it may hide the cases that are important for the diagnostician.

In the next section we propose another way to consider the space of diagnoses that considers the required use of the diagnoses.

4.4 Use equivalence of diagnoses

As discussed in the introduction, the diagnostic process has a specific purpose. Hence, the definition of “a diagnosis” should enable the diagnostic process to fulfill its purpose. Of the many possible purposes of a diagnosis, this section assumes the purpose of a diagnosis to be both treatment and provision of an explanation for the abnormalities.

The idea to be developed is that a diagnosis corresponds to a formula such that all of the models of the formula are use equivalent. Use equivalence of a set of interpretations means that the action taken is independent of which of the interpretations is true.

Here the “use” can include both descriptions of what is wrong, and treatments. A set of interpretations could be use-equivalent if the both would have the same description (e.g., “minor infection”, or “cpu board is shot”) and the same treatment (e.g., “give penicillin” or “replace whole cpu board”). If the interpretations differ in the description or in the treatment they are not use-equivalent.

The rationale is that there is no point in distinguishing between use-equivalent interpretations; as far as the diagnosis is concerned use-equivalent interpretations should be considered as the same diagnosis. The aim of diagnosis is to refine the set of use-equivalent possibilities. The use-equivalence induces a partition on the set of interpretations. Each of these partitions corresponds to a possible distinct diagnosis.

Let T be the set of all treatments. By a treatment we mean a total prescription of what to do (i.e., we do not conjoin different treatments — the conjunction would be one treatment).

Let \mathfrak{I} be the set of all interpretations for our language⁶ [Chang and Lee, 1973].

Definition 4.4 The possible treatment space⁷ \mathcal{P} is a subset of $\mathfrak{I} \times T$. $\langle I, T \rangle \in \mathcal{P}$ means that T is a possible treatment given that the world is denoted by I .

\mathcal{P} induces an equivalence relation on the set of interpretations. This will be called *strong equivalence* with respect to \mathcal{P} . The idea is that equivalent interpretations have the same set of possible treatments. In section 4.5 this will be contrasted with a “weak equivalence” that ensures that there is one treatment that covers the set of interpretations.

Definition 4.5 Two interpretations I_1 and I_2 are strongly equivalent with respect to \mathcal{P} , written $I_1 \equiv_{\mathcal{P}} I_2$ if $\forall T \in T$, $\langle I_1, T \rangle \in \mathcal{P}$ if and only if $\langle I_2, T \rangle \in \mathcal{P}$.

The following lemma can be simply verified:

Lemma 4.6 $\equiv_{\mathcal{P}}$ is an equivalence relation.

We will also say two formulae are equivalent with respect to \mathcal{P} if all of the models of all of the formulae are equivalent with respect to \mathcal{P} .

This equivalence relation induces a partition on the space of interpretations. The diagnosis space of use-diagnoses is the space D of formulae that describe⁸ the partitions induced by strong equivalence with respect to \mathcal{P} . As outlined in section 4.1, defining the diagnosis space completely determines a use-diagnosis.

⁶Note that we never actually represent interpretations. They are considered to be all the ways that the world under consideration (the system being diagnosed) could be.

⁷Section 4.6 considers a measure (over the space of possible treatments) of how good the treatment is; this corresponds to a utility measure. This definition is deliberately left open to include (1) “optimising diagnoses” where the relation is between interpretations and the optimal diagnosis, and (2) “satisficing diagnoses” where the relation is between interpretations and possible satisfactory treatments.

⁸We assume that the partitions are finitely describable. This does not seem to be a severe practical restriction.

Definition 4.7 D is a possible use-diagnosis with respect to possible treatments \mathcal{P} if and only if the models of D form a maximal (with respect to set inclusion) element of S , where S is given by

$$\{S : S \subseteq \mathfrak{I}, \forall T \in T \forall I_1 \in S \forall I_2 \in S \\ \langle I_1, T \rangle \in \mathcal{P} \Rightarrow \langle I_2, T \rangle \in \mathcal{P}\}$$

Definition 4.8 A use-diagnosis is a possible use diagnosis that is consistent with $SD \cup OBS$.

What is important about the set of use-diagnoses is that they convey all information about the system necessary for providing treatment.

The implicit assumption of [de Kleer et al., 1990], that the set of diagnoses conveys all of the information about the system that may be useful for diagnosis, can be summed up as:

Assumption 4.9 [de Kleer et al., 1990]: The models of each possible diagnosis (i.e., truth assignments to formulae $AB(C)$) are strongly equivalent with respect to possible treatments \mathcal{P} .

That is, the set of truth assignments forms a partition no coarser than the space of use-diagnoses. Although they are implicitly using this assumption in [de Kleer et al., 1990], they do not allow the use of this equivalence to form higher levels of diagnosis.

Note that, while the disjunct of kernel diagnoses is logically equivalent to the set of all diagnoses, the set of all use-equivalent diagnoses need not imply the set of all diagnoses.

Example 4.10 When building partial diagnoses we are combining descriptions of the form

$$A \wedge B \\ A \wedge \neg B$$

into one class that is labelled with A .

If B was (possibly) important to determine treatment, then the diagnoses $A \wedge B$ and $A \wedge \neg B$ should not be combined into one equivalence class, even though they are both possible given the observations. This will be important for a differential diagnosis. It may be important to devise a test to distinguish between the two diagnoses.

The other case is when B is unimportant to determine use when A is true. In this case we would want to combine these diagnoses into one equivalence class. Note that B may still be important when A is not true.

Example 4.11 Suppose

$$AB(A) \wedge AB(B) \wedge \neg AB(C) \\ AB(A) \wedge AB(B) \wedge \neg AB(C) \\ AB(A) \wedge \neg AB(B) \wedge AB(C)$$

are all in the same equivalence class; i.e., each formula has the same use (treatment/description). Suppose that each of these is consistent with the observations. Consider

$$D_4 = AB(A) \wedge \neg AB(B) \wedge \neg AB(C)$$

to be a much more serious condition that would require different treatment. The use-diagnoses can be contrasted with kernel-diagnoses by considering the cases of whether D_4 is inconsistent with $SD \cup OBS$:

1. Suppose D_4 is inconsistent with $SD \cup OBS$. In this case, there is one use-diagnosis, i.e. there is only one remaining treatment/description. There are however two kernel diagnoses, namely

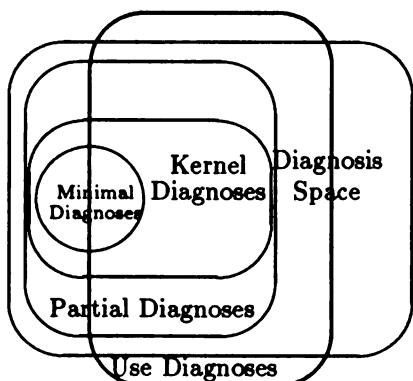
$$\{AB(A) \wedge AB(B), AB(A) \wedge AB(C)\}.$$

These correspond to trying to find a statement that allows description of the situation as concisely as possible.

2. Suppose D_4 is consistent with $SD \cup OBS$. In this case, there are two use-diagnoses, i.e. there are two different sets of treatments that may need to be distinguished by a differential diagnosis. There is, however, only one kernel diagnosis, namely $AB(A)$.

Figure 4 summarises the relationships among the different equivalence classes discussed in this section.

Figure 4: Diagnostic Equivalence Classes



Note that the use diagnoses are not a strict subset of the diagnoses. The diagnoses only represent the space of values of sentences containing AB-predicates. The use diagnoses are not restricted to this space. There may be some other predicates that are important for treatment that the treatment that is not conveyed by the normality and abnormality of components (e.g., allergies or the ease of access of some component may affect our treatment).

This figure also points out that consistency-based diagnoses miss some part of a larger space of diagnoses

important for determining treatment, i.e. the space of diagnoses C_{CB} loses information. This is the point of assumption 4.9. If this assumption is violated, then the use-diagnoses can have elements that do not intersect with the space of "diagnoses".

4.5 Weakly equivalent interpretations

The previous definition of a use-diagnosis assumed that there was no difference in the treatments for the different models of the diagnosis. That is, any treatment for one interpretation is a treatment for the others. While this may define a space of diagnoses that are in some sense equivalent, it seems to be too strong for practical diagnosis where we only have to carry out one (perhaps complex) treatment. Grouping interpretations, if there is a treatment that covers them all, is the goal.

Interpretations can be grouped to form a **weak use-diagnosis** if there exists a treatment for every interpretation in the class:

Definition 4.12 Two interpretations I_1 and I_2 are **weakly use-equivalent** with respect to possible treatment space \mathcal{P} if $\exists T \in \mathcal{T}$ such that $\langle I_1, T \rangle \in \mathcal{P}$ and $\langle I_2, T \rangle \in \mathcal{P}$.

Definition 4.13 A **weak use-diagnoses** is defined as the formulae D such that all of the models of D are weakly use-equivalent with respect to some $T \in \mathcal{T}$.

It makes sense to keep testing until we have formed a single weak use-diagnosis, and not use a test unless it makes progress towards finding a unique weak use-diagnosis.

There are a number of refinements that can be made to the notion of a weak use-diagnosis to make it more useful:

1. As the definition stands, the weak use-diagnoses are not disjoint. This could lead us to a problem in naively applying a probability measure over diagnoses (as we can't just add the probabilities of the diagnoses to determine the probability of a disjunct of a set of diagnoses). It does not give us the same problems that were stated for partial and kernel diagnoses in section 4.3. First, the distinctions that are hidden in a single diagnosis are not important if we apply the treatment that unifies the diagnosis. Second, it makes sense to apply tests to distinguish sets of weak-use diagnoses, as these tests are really tests to determine what treatment to apply.
2. Identifying use diagnoses will require at most as much computational effort as identifying kernel/minimal diagnoses, and in many cases it will require less effort. This is due to the difference in

how fine a partitioning of the diagnosis space is required. Identifying kernel/minimal diagnoses is a computationally demanding task (it contains NP-hard subtasks), as this approach precisely identifies the diagnosis. This entails a very fine partitioning of the diagnosis space. In contrast, the description of a use-diagnosis does not always necessitate the computation of a diagnosis on such a fine level (e.g. computing the intersection of the use equivalent partitions and the models of $SD \cup OBS$); the partitioning of the space is only as fine as will distinguish the intended uses of the diagnoses. At worst, this partitioning will be as fine as that of the kernel/minimal diagnoses. In this latter approach, making fine distinctions facilitates the diagnostic task only if such distinctions are important to the intended use of the diagnosis.

For example, a patient may have a diagnosis that can be described as “liver disease”. This does not mean that the patient could have any possible liver disease, but rather that any finer distinctions would not provide any useful information for further tests of treatment. Sometimes the diagnosis of “liver disease” may be more precisely defined to exclude the sub-classes of liver disease that would involve a different course of tests or treatment. In both cases it may be unnecessary to specify a logical formula that completely describes the space of possibilities.

In example 4.11, we can use the diagnosis $AB(A)$ to describe the diagnosis in the first case, even though not all diagnoses that can be derived from $AB(A)$ are possible⁹.

4.6 Probability, Utility and Influence diagrams

In this section we show how the previous notion of a use-diagnosis can be related to classical decision theory. In classical decision theory (e.g., [Berger, 1985]), we assume we have a measure $\mu(I, T)$ of the utility of treatment T in interpretation (or possible world) I . The expected value, $\mathcal{E}(T)$, of treatment T is defined by

$$\mathcal{E}(T) = \sum_{I \in \mathfrak{S}} \mu(I, T) \times p(I)$$

We can define the possible treatment space as the set of interpretations with the same utility¹⁰. In this case, “strong use-equivalence” means having the same utility for each treatment.

⁹This is similar to the fact that not all of the models of a diagnosis of [de Kleer et al., 1990] are possible.

¹⁰Formally, the treatment in the possible treatment space would be a pair (T, v) where $(I, (T, v)) \in \mathcal{P}$ if $\mu(I, T) = v$.

Let \mathcal{D} be the set of use-diagnoses. For $D \in \mathcal{D}$, every model of D has the same utility measure; let $\mu(D, T)$ be the value of $\mu(I, T)$ for some interpretation I such that D is true in I .

Proposition 4.14

$$\mathcal{E}(T) = \sum_{D \in \mathcal{D}} \mu(D, T) \times p(D)$$

Proof:

$$\begin{aligned} \mathcal{E}(T) &= \sum_{I \in \mathfrak{S}} \mu(I, T) \times p(I) \\ &= \sum_{D \in \mathcal{D}} \sum_{I \models D} \mu(I, T) \times p(I) \\ &= \sum_{D \in \mathcal{D}} \mu(D, T) \times \sum_{I \models D} p(I) \\ &= \sum_{D \in \mathcal{D}} \mu(D, T) \times p(D) \end{aligned}$$

□

This tells us that considering just the use-diagnoses is all we need to determine treatment. This should not be surprising, as that was the intention behind the definition.

5 CONCLUSIONS

This paper has argued that the complete diagnostic cycle is necessary for the notion of diagnosis to make any sense. In addition it has shown that consistency-based definitions of diagnosis (1) are defined according to logical notions of consistent entailment; (2) require extra-logical criteria to compute minimal, kernel or most likely diagnoses; (3) do not make use-free or “first principles” definitions of minimal, kernel or most likely diagnosis.

This paper also questions a definition of a consistency-based diagnosis as being “a first principles” or fundamental definition. We argue that this approach is inadequate to provide the “specification for an ideal diagnostician” [de Kleer et al., 1990, p.329]; the fundamental notion of diagnosis is one of utility, not logic-based consistency. Once the use of the diagnosis has been determined and a utility function defined, then the form of the diagnosis can be computed.

A more general framework, that of partitioning the space of possible diagnoses into equivalence classes of diagnoses, has been discussed. Consistency-based diagnoses are shown to be one class of diagnoses, a class which is warranted only under certain conditions. Another class, that of use-equivalent diagnoses, is proposed as a better means of computing diagnoses. Such a class minimises computational expense by computing the coarsest possible partition of the search space

to enable an appropriate treatment. A use-equivalent diagnosis reduces to a consistency-based diagnosis in the extreme case when "the diagnosis" needs to be precisely defined using the notion of logical consistency.

References

- [Berger, 1985] J.O. Berger. *Statistical Decision Theory*. Springer Verlag, 1985.
- [Chang and Lee, 1973] C. Chang and R.C. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, NY, 1973.
- [Cheeseman, 1990] P. Cheeseman. On finding the most probable model. In J. Shranger and P. Langley, editors, *Computational Models of Scientific Discovery and Theory Formation*, chapter 3, pages 73–95, Morgan Kaufmann, San Mateo, 1990.
- [de Kleer and Williams, 1987] J. de Kleer and B. Williams. Diagnosing Multiple Faults. *Artificial Intelligence*, 32:97–130, 1987.
- [de Kleer et al., 1990] J. de Kleer, A. Mackworth, and R. Reiter. Characterizing Diagnoses. In *Proc. AAAI*, pages 324–330, 1990.
- [Elstein et al., 1978] A. Elstein, L. Shulman, and S. Sprafka. *Medical Problem Solving: An Analysis of Clinical Problem Solving*. Harvard University Press, Cambridge, MA, 1978.
- [Genesereth, 1984] M. Genesereth. The Use of Design Descriptions in Automated Diagnosis. *Artificial Intelligence*, 41:1–436, 1984.
- [Heckerman and Horvitz, 1990] D. Heckerman and E. Horvitz. Problem Formulation as the Reduction of a Decision Model. In *Proc. Conf. Uncertainty in Artificial Intelligence*, pages 82–89, 1990.
- [Kean and Tsiknis, 1990] A. Kean and G. Tsiknis. An Incremental Method for Generating Prime Implicants/Implicates. *Journal of Symbolic Computation*, 9:185–206, 1990.
- [Kohavi, 1978] A. Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, 1978.
- [Ledley and Lusted, 1959] R.S. Ledley and L.B. Lusted. Reasoning Foundations of Medical Diagnosis. *Science*, 130:9–21, 1959.
- [Lin and Goebel, 1990] D. Lin and R. Goebel. The Probability of Causal Explanations. In *Proc. CSCSI-90*, pages 15–20, 1990.
- [McCarthy, 1986] J. McCarthy. Applications of Circumscription to Formalizing Commonsense Knowledge. *Artificial Intelligence Journal*, 28:89–116, 1986.
- [McCluskey, 1956] E.J. McCluskey. Minimization of Boolean Functions. *Bell System Technical J.*, 35:1417–1444, 1956.
- [Pearl, 1986] J. Pearl. Fusion, Propagation, and Structuring in Belief Networks. *Artificial Intelligence*, 29:241–288, 1986.
- [Pearl, 1987] J. Pearl. Distributed Revision of Composite Beliefs. *Artificial Intelligence*, 33:173–215, 1987.
- [Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
- [Peng and Reggia, 1987a] Y. Peng and J. Reggia. A Probabilistic Causal Model for Diagnostic Problem Solving. Part Two: Diagnostic Strategy. *IEEE Trans. Systems, Man and Cybernetics*, 17:395–406, 1987.
- [Peng and Reggia, 1987b] Y. Peng and J. Reggia. A Probabilistic Causal Model for Diagnostic Problem Solving. Part One: Integrating Symbolic Causal Inference with Numeric Probabilistic Inference. *IEEE Trans. Systems, Man and Cybernetics*, 17:146–162, 1987.
- [Peng and Reggia, 1987c] Y. Peng and J. Reggia. Being Comfortable with Plausible Diagnostic Hypotheses. Technical Report TR-1753, Dept. of Computer Science, University of Maryland, 1987.
- [Poole, 1988] D.L. Poole. Representing Knowledge for Logic-based Diagnosis. In *Proc. Int'l. Conf. on Fifth Generation Computer Systems*, pages 1282–1290, 1988.
- [Poole, 1989a] D.L. Poole. Explanation and Prediction: An Architecture for Default and Abductive Reasoning. *Computational Intelligence*, 5:97–110, 1989.
- [Poole, 1989b] D.L. Poole. Normality and Faults in Logic-based Diagnosis. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1304–1310, 1989.
- [Poole and Provan, 1990] D. Poole and G.M. Provan. What is an Optimal Diagnosis? In *Proc. Conf. Uncertainty in Artificial Intelligence*, pages 46–53, 1990.
- [Pople, 1982] H. Pople. Heuristic Methods for Imposing Structure on Ill-Structured Problems: The Structuring of Medical Diagnoses. In P. Szolovits, editor, *AI in Medicine*, pages 119–190, Westview Press, 1982.
- [Provan, 1990] G. Provan. The Computational Complexity of Multiple-Context Truth Maintenance Systems. In *Proc. ECAI*, pages 522–527, 1990.
- [Reggia et al., 1985] J. Reggia, D. Nau, and P. Wang. A Formal Model of Diagnostic Inference. *Information Sciences*, 35:227–285, 1985.
- [Reiter, 1987] R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32:57–96, 1987.
- [Shortliffe and Perrault, 1990] E.H. Shortliffe and L.E. Perrault, editors. *Medical Informatics*. Addison-Wesley, 1990.

Modeling Rational Agents within a BDI-Architecture*

Anand S. Rao
 Australian Artificial Intelligence Institute
 Carlton, Victoria 3053
 Australia
 Email: anand@aaii.oz.au

Michael P. Georgeff
 Australian Artificial Intelligence Institute
 Carlton, Victoria 3053
 Australia
 Email: georgeff@aaii.oz.au

Abstract

Intentions, an integral part of the mental state of an agent, play an important role in determining the behavior of rational agents as they seek to attain their goals. In this paper, a formalization of intentions based on a branching-time possible-worlds model is presented. It is shown how the formalism realizes many of the important elements of Bratman's theory of intention. In particular, the notion of intention developed here has equal status with the notions of belief and desire, and cannot be reduced to these concepts. This allows different types of rational agents to be modeled by imposing certain conditions on the persistence of an agent's beliefs, goals, and intentions. Finally, the formalism is compared with Bratman's theory of intention and Cohen and Levesque's formalization of intentions.

1 INTRODUCTION

The role played by attitudes such as beliefs (B), desires (D) (or goals (G)), and intentions (I) in the design of rational agents has been well recognized in the philosophical and AI literature [2, 3, 9]. Systems and formalisms that give primary importance to intentions are often referred to as BDI-architectures. While most philosophical theories treat intentions as being reducible to beliefs and desires, Bratman argues convincingly that intentions play a significant and distinct role in practical reasoning. He treats intentions as partial plans of action that the agent is committed to execute to fulfill her goals.

Some of the philosophical aspects of Bratman's theory were formalized by Cohen and Levesque [6]. In their

formalism, intentions are defined in terms of temporal sequences of an agent's beliefs and goals. In particular, an agent *fanatically committed* to her intentions will maintain her goals until either they are believed to be achieved or believed to be unachievable; an agent with a *relativized commitment* to her intentions is similarly committed to her goals but may also drop them when some specified conditions are believed to hold.

In this paper, we present an alternative possible-worlds formalism for BDI-architectures. There are three crucial elements to the formalism. First, intentions are treated as first-class citizens on a par with beliefs and goals. This allows us to define different strategies of commitment with respect to an agent's intentions and thus to model a wide variety of agents. Second, we distinguish between the *choice* an agent has over the actions she can perform and the *possibilities* of different outcomes of an action. In the former case, the agent can choose among outcomes; in the latter case, the environment makes that determination. Third, we specify an interrelationship between beliefs, goals, and intentions that allows us to avoid many of the problems usually associated with possible-worlds formalisms, such as commitment to unwanted side effects.

In the following sections, we briefly outline the formalism and describe some of its more important features. We then define a number of different commitment strategies and show how these affect agent behavior.

2 INFORMAL SEMANTICS

We choose to model the world using a temporal structure with a branching time future and a single past, called a *time tree*. A particular time point in a particular world is called a *situation*.

Event types transform one time point into another. Primitive events are those events directly performable by the agent and uniquely determine the next time point in a time tree. Non-primitive events map to non-adjacent time points, thus allowing us to model

*This research was partly supported by a Generic Industrial Research and Development Grant from the Department of Industry, Technology and Commerce, Australia.

the partial nature of plans. Their potential for decomposition into primitive events can be used to model hierarchical plan development.

The branches in a time tree can be viewed as representing the *choices* available to the agent at each moment of time. For example, if there are two branches emanating from a particular time point, one labeled e_1 , say, and the other e_2 , then the agent has a choice of executing e_1 and moving to the next time point along the branch of the time tree labeled with e_1 , or of executing e_2 and likewise moving along its associated branch.

Of course, the agent may attempt to execute some event, but fail to do so. We thus distinguish between the successful execution of events and their failure and label the branches accordingly. As we shall see later, this distinction is critical in having an agent act on her intentions without requiring her to be successful in her attempts.

We use a formalism similar to Computation Tree Logic, CTL*, [7] to describe these structures.¹ A distinction is made between *state formulas* and *path formulas*: the former are evaluated at a specified time point in a time tree and the latter over a specified path in a time tree. We introduce two modal operators, *optional* and *inevitable*, which operate on path formulas. A path formula ψ is said to be *optional* if, at a particular time point in a time tree, ψ is true of at least one path emanating from that point; it is *inevitable* if ψ is true of all paths emanating from that point.² The standard temporal operators \bigcirc (next), \Diamond (eventually), \Box (always), and U (until), operate over state and path formulas.

These modalities can be combined in various ways to describe the options available to the agent, such as shown in Figure 1. For example, the structure shown in the figure could be used to represent the following statements: it is *optional* that John will *eventually* visit London (denoted by p); it is *optional* that Mary will *always* live in Australia (r); it is *inevitable* that the world will *eventually* come to an end (q); and it is *inevitable* that one plus one will *always* be two (s).

Belief is modeled in the conventional way. That is, in each situation we associate a set of *belief-accessible* worlds; intuitively, those worlds that the agent *believes* to be possible. Unlike most conventional models of belief, however, each belief-accessible world is a time tree. Multiple belief-accessible worlds result from the agent's lack of knowledge about the state of the world. But within each of these worlds, the branching future represents the choice (options) still available to the agent in selecting which actions to perform.

Further insight into the approach is provided by com-

¹Elsewhere [15] we use an explicit notion of time to describe these structures.

²In CTL*, E and A are used to denote these operators.

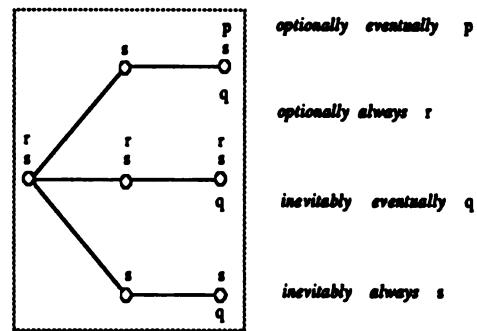


Figure 1: Temporal modalities

paring the above possible-worlds model with a conventional decision tree. In this case, each arc emanating from a *chance* node of a decision tree corresponds to a possible world, and each arc emanating from a *decision* node to the choice available within a possible world. A formal comparison of our possible-worlds model with the decision-tree representation is carried out elsewhere [14].

Similar to belief-accessible worlds, for each situation we also associate a set of *goal-accessible* worlds to represent the goals of the agent. Although, in the general case, desires can be inconsistent with one another, we require that goals be consistent. In other words, goals are chosen desires of the agent that are consistent. Moreover, the agent should believe that the goal is achievable. This prevents the agent from adopting goals that she believes are unachievable and is one of the distinguishing properties of goals as opposed to desires. Cohen and Levesque [5] call this the property of *realism*.

In this paper, we adopt a notion of *strong realism*. In particular, we require that the agent believe she can optionally achieve her goals, by carefully choosing the events she executes (or, more generally, that get executed by her or any other agent). We enforce this notion of compatibility by requiring that, for each belief-accessible world w at a given moment in time t , there must be a goal-accessible world that is a *sub-world* of w at time t . Figure 2 illustrates this relation between belief- and goal-accessible worlds. The goal-accessible world $g1$ is a sub-world of the belief-accessible world $b1$.

Intentions are similarly represented by sets of *intention-accessible* worlds. These worlds are ones that the agent has *committed* to attempt to realize. Similar to the requirement for belief-goal compatibility, the intention-accessible worlds of the agent must be compatible with her goal-accessible worlds; an agent can only intend some course of action if it is one of her goals. Consequently, corresponding to each

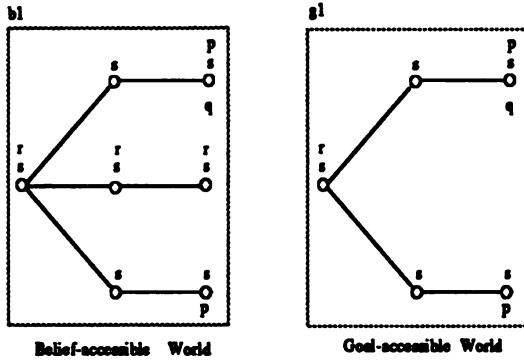


Figure 2: Subworld relationship between beliefs and goals

goal-accessible world w at time t , there must be an intention-accessible world that is a *sub-world* of w at time t . Intuitively, the agent chooses some course of action in w and commits herself to attempt its execution.

In this framework, different belief-, goal-, and intention-accessible worlds represent different possible scenarios for the agent. Intuitively, the agent believes the actual world to be one of her belief-accessible worlds; if it were to be belief world $b1$, then her goals (with respect to $b1$) would be the corresponding goal-accessible world, $g1$ say, and her intentions the corresponding intention-accessible world, $i1$. As mentioned above, $g1$ and $i1$ represent increasingly selective choices from $b1$ about the desire for and commitment to possible future courses of action.

While for every belief-accessible world there must be a goal-accessible world (and similarly for intentions), the converse need not hold. Thus, even if the agent believes that certain facts are inevitable, she is not forced to adopt them as goals (or as intentions). This means that goals and intentions, while having to be consistent, need not be closed under the beliefs of the agent.

In this way, an agent believing that it is inevitable that pain (p) always accompanies having a tooth filled (f), may yet have the goal (or intention) to have a tooth filled without also having the goal (or intention) to suffer pain. This relationship between belief, goal, and intention-accessible worlds is illustrated by the example shown in Figure 3. Although the agent believes that *inevitably always* ($f \supset p$), she does not adopt this as a goal nor as an intention. Similarly, although the agent adopts the goal (and intention) to achieve f , she does not thereby acquire the goal (or intention) p .

The semantics of beliefs, goals, and intentions given above is formalized in Section 3. It thus remains to be shown how these attitudes determine the actions of

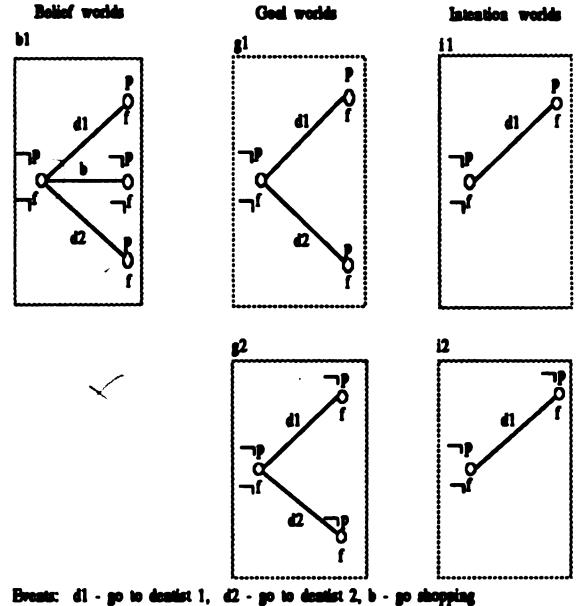


Figure 3: Belief, Goal, and Intention Worlds

an agent and how they are formed, maintained, and revised as the agent interacts with her environment. Different types of agent will have different schemes for doing this, which in turn will determine their behavioral characteristics. We consider some of these schemes and their formalization in Section 4.

3 FORMAL THEORY

3.1 SYNTAX

CTL* [7] is a propositional branching-time logic used for reasoning about programs. We extend this logic in two ways. First, we describe a first-order variant of the logic. Second, we extend this logic to a possible-worlds framework by introducing modal operators for beliefs, goals, and intentions. While Emerson and Srinivasan [7] provide a sound and complete axiomatization for their logic, we do not address the issue of completeness in this paper. Our main aim is to present an expressive semantics for intentions and to investigate certain axioms that relate intentions to beliefs and goals within this structure.

Similar to CTL*, we have two types of formulas in our logic: *state formulas* (which are evaluated at a given time point in a given world) and *path formulas* (which are evaluated along a given path in a given world). A state formula can be defined as follows:

- any first-order formula is a state formula;

- if ϕ_1 and ϕ_2 are state formulas and x is an individual or event variable, then $\neg\phi_1$, $\phi_1 \vee \phi_2$, and $\exists x \phi_1(x)$ are state formulas;
- if e is an event type then $succeeds(e)$, $fails(e)$, $does(e)$, $succeeded(e)$, $failed(e)$, and $done(e)$ are state formulas;
- if ϕ is state formula then $BEL(\phi)$, $GOAL(\phi)$ and $INTEND(\phi)$ are state formulas; and
- if ψ is a path formula, then $optional(\psi)$ is a state formula.

A path formula can be defined as follows:

- any state formula is also a path formula; and
- if ψ_1 and ψ_2 are path formulas, then $\neg\psi_1$, $\psi_1 \vee \psi_2$, $\psi_1 \cup \psi_2$, $\Diamond\psi_1$, $\Box\psi_1$ are path formulas.

Intuitively, the formulas $succeeded(e)$ and $failed(e)$ represent the immediate past performance, respectively successfully and unsuccessfully, of event e . The formula $done(e)$ represents the immediate past occurrence of e , either successfully performed or not. The formulas $succeeds(e)$, $fails(e)$, and $does(e)$ are similarly defined but refer to the immediate future occurrence of events. The operators BEL , $GOAL$, and $INTEND$ represent, respectively, the beliefs, goals, and intentions of the agent.

3.2 POSSIBLE-WORLDS SEMANTICS

We first provide the semantics of various state and path formulas. This will be followed by the semantics of events and, finally, the possible-worlds semantics of beliefs, goals, and intentions.

Definition 1 : An interpretation M is defined to be a tuple, $M = \langle W, E, T, \prec, U, \mathcal{B}, \mathcal{G}, \mathcal{I}, \Phi \rangle$. W is a set of worlds, E is a set of primitive event types, T is a set of time points, \prec a binary relation on time points,³ U is the universe of discourse, and Φ is a mapping of first-order entities to elements in U for any given world and time point. A situation is a world, say w , at a particular time point, say t , and is denoted by w_t . The relations, \mathcal{B} , \mathcal{G} , and \mathcal{I} map the agent's current situation to her belief, goal, and intention-accessible worlds, respectively. More formally, $\mathcal{B} \subseteq W \times T \times W$ and similarly for \mathcal{G} and \mathcal{I} . Sometimes we shall use \mathcal{R} to refer to any one of these relations and shall use \mathcal{R}_t^w to denote the set of worlds \mathcal{R} -accessible from world w at time t . Figure 4 shows how the belief relation \mathcal{B} maps the world w_0 at time t_1 to the worlds $b1$ and $b2$. In other words, $\mathcal{B}_{t_1}^{w_0} = \{b1, b2\}$.

Definition 2 : Each world w of W , called a *time tree*, is a tuple $\langle T_w, \mathcal{A}_w, \mathcal{S}_w, \mathcal{F}_w \rangle$, where $T_w \subseteq T$ is a set

³We require that the binary relation be total, transitive and backward-linear to enforce a single past and branching future.

of time points in the world w and \mathcal{A}_w is the same as \prec , restricted to time points in T_w . A *fullpath* in a world w is an infinite sequence of time points (t_0, t_1, \dots) such that $\forall i (t_i, t_{i+1}) \in \mathcal{A}_w$. We use the notation $(w_{t_0}, w_{t_1}, \dots)$ to make the world of a particular fullpath explicit. The arc functions \mathcal{S}_w and \mathcal{F}_w map adjacent time points to events in E . More formally, $\mathcal{S}_w: T_w \times T_w \rightarrow E$ and similarly for \mathcal{F}_w . We require that if $\mathcal{S}_w(t_i, t_j) = S_w(t_i, t_k)$, then $t_j = t_k$ and similarly for \mathcal{F}_w . Also, the domains of \mathcal{S}_w and \mathcal{F}_w are disjoint. Intuitively, for any two adjacent time points for which the arc function \mathcal{S}_w is defined, its value represents the event that successfully occurred between those time points. Similarly, the value of the arc function \mathcal{F}_w represents the failure of events occurring between adjacent time points.

Definition 3 : A *sub-world* is defined to be a sub-tree of a world with the same truth-assignment of formulas. A world w' is a *sub-world* of the world w , denoted by $w' \sqsubseteq w$, if and only if (a) $T_{w'} \subseteq T_w$; (b) for all $u \in T_{w'}$, $\Phi(q, w', u) = \Phi(q, w, u)$, where q is a predicate symbol; (c) for all $u \in T_{w'}$, $\mathcal{R}_u^w = \mathcal{R}_u^{w'}$; and (d) $\mathcal{A}_{w'}$ is \mathcal{A}_w restricted to time points in $T_{w'}$ and similarly for $\mathcal{S}_{w'}$ and $\mathcal{F}_{w'}$. We say that w' is a *strict sub-world* of w denoted by $w' \subset w$ if and only if $w' \sqsubseteq w$ and $w \not\sqsubseteq w'$.

Now consider an interpretation M , with a variable assignment v .⁴ We take v_d^i to be that function that yields d for the variable i and is the same as v everywhere else. The semantics of first-order formulas can be given as follows:

- $M, v, w_t \models q(y_1, \dots, y_n)$ iff $\langle v(y_1), \dots, v(y_n) \rangle \in \Phi[q, w, t]$ where $q(y_1, \dots, y_n)$ is a predicate formula.
- $M, v, w_t \models \neg\phi$ iff $M, v, w_t \not\models \phi$.
- $M, v, w_t \models \phi_1 \vee \phi_2$ iff $M, v, w_t \models \phi_1$ or $M, v, w_t \models \phi_2$.
- $M, v, w_t \models \exists i \phi$ iff $M, v_i^d, w_t \models \phi$ for some d in U .
- $M, v, (w_{t_0}, w_{t_1}, \dots) \models \phi$ iff $M, v, w_{t_0} \models \phi$.
- $M, v, (w_{t_0}, w_{t_1}, \dots) \models \Box\psi$ iff $M, v, (w_{t_1}, \dots) \models \psi$.
- $M, v, (w_{t_0}, w_{t_1}, \dots) \models \Diamond\psi$ iff $\exists k, k \geq 0$ such that $M, v, (w_{t_k}, \dots) \models \psi$.
- $M, v, (w_{t_0}, w_{t_1}, \dots) \models \psi_1 \cup \psi_2$ iff
 - (a) $\exists k, k \geq 0$ such that $M, v, (w_{t_k}, \dots) \models \psi_2$ and for all $0 \leq j < k$, $M, v, (w_{t_j}, \dots) \models \psi_1$ or
 - (b) for all $j \geq 0$, $M, v, (w_{t_j}, \dots) \models \psi_1$.
- $M, v, w_{t_0} \models optional(\psi)$ iff there exists a fullpath $(w_{t_0}, w_{t_1}, \dots)$ such that $M, v, (w_{t_0}, w_{t_1}, \dots) \models \psi$.

The formula *inevitable*(ψ) is defined as $\neg optional(\neg\psi)$ and $\Box\psi$ is defined as $\neg\Diamond\neg\psi$. The definition of U (until) given above is that of *weak until*, which allows fullpaths in which ψ_1 is true forever. Well-formed formulas that contain no positive occurrences of *inevitable* (or negative occurrences of *optional*) outside the scope of the

⁴For the sake of simplicity, we shall assume that the variable assignment of event terms are events denoted by the same letter, i.e., $v(e) = e$ for any event term e .

modal operators **BEL**, **GOAL**, or **INTEND** will be called O-formulas and denoted by α . Conversely, we define I-formulas, denoted by β , to contain no positive occurrences of *optional*.

3.2.1 Semantics of Events

Event types transform one time point into another. The various aspects involved in this transformation are called the *dynamics* of a system [8, 13]. Just as one can define the *truth* or *falsity* of formulas at a time point, we need mechanisms for defining the *success* or *failure* of events in transforming one time point to another.

We use the formula *succeeded*(e) to denote the successful execution of event e by the agent, and *failed*(e) to denote its failure. Note that event e *not occurring* is not the same as the event e *fails*. Failure of event types alter the world irrevocably, possibly forcing the agent to replan or revise her plans. This aspect is crucial in capturing the dynamics of any system. For example, the consequences of a thief successfully robbing a bank is quite different from the thief failing in his attempt to rob the bank, which is again different from the thief not attempting to rob the bank. All three are distinct behaviors and have to be distinguished accordingly.

We say that the agent has *done*(e) if she has either *succeeded* or *failed* in doing the event. The notions *succeeds*, *failed*, and *does* are similarly defined, but require the event to occur on all paths emanating from the time point at which the formula is evaluated.

More formally, we have:

$M, v, w_{t_1} \models \text{succeeded}(e)$ iff there exists t_0 such that $S_w(t_0, t_1) = e$.

$M, v, w_{t_1} \models \text{failed}(e)$ iff there exists t_0 such that $F_w(t_0, t_1) = e$.

The formula *done*(e) is defined as

$\text{succeeded}(e) \vee \text{failed}(e)$; *succeeds*(e) is defined as *inevitable* $\bigcirc(\text{succeeded}(e))$; *fails*(e) is defined as *inevitable* $\bigcirc(\text{failed}(e))$; *does*(e) is defined as *inevitable* $\bigcirc(\text{done}(e))$;

In this paper, we have considered only single-agent, non-parallel actions. If parallel actions among multiple agents are to be allowed, the functions S_w and F_w must be extended to map to a set of event-agent pairs, signifying which events are performed by which agents.

3.2.2 Semantics of Beliefs, Goals, and Intentions

The traditional possible-worlds semantics of beliefs considers each world to be a collection of propositions and models belief by a belief-accessibility relation \mathcal{B} linking these worlds. A formula is said to be believed in a world if and only if it is true in all its belief-

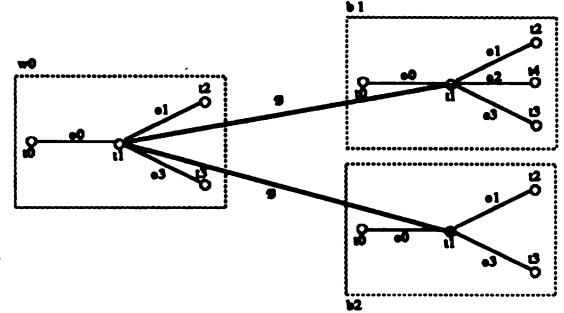


Figure 4: Worlds as time trees

accessible worlds [10].

Cohen and Levesque [6] treat each possible world as a *time-line* representing a sequence of events, temporally extended infinitely into the past and the future. As discussed in Section 2, we instead consider each possible world to be a *time tree*. Each time tree denotes the optional courses of events choosable by an agent in a particular world. The belief relation maps a possible world at a time point to other possible worlds. We say that an agent has a belief ϕ , denoted **BEL**(ϕ), at time point t if and only if ϕ is true in all the belief-accessible worlds of the agent at time t .

Figure 4 shows how the belief relation \mathcal{B} maps the world w_0 at time t_1 to the worlds b_1 and b_2 . Let us assume that the formulas that are true at t_1 in b_1 are ϕ_1 and ϕ_2 , while the formulas that are true at t_1 in b_2 are ϕ_1 and $\neg\phi_2$. From this it is easy to conclude that **BEL**(ϕ_1) and $\neg\text{BEL}(\phi_2)$ are true at t_1 in w_0 . As discussed earlier, ϕ_1 and ϕ_2 could be any state formulas; in particular, ones involving the future options available to the agent.

As the belief relation is time-dependent, the mapping of \mathcal{B} at some other time point, say t_2 , may be different from the one at t_1 . Thus the agent can change her beliefs about the options available to her.

The semantics of the modal operator **GOAL** is given in terms of a goal-accessible relation \mathcal{G} which is similar to that of the \mathcal{B} relation. The goal-accessibility relation specifies situations that the agent *wishes* to be in. Thus, in the same way that we treat belief, we say that the agent has a goal ϕ at time t if and only if ϕ is true in all the goal-accessible worlds of the agent at time t .

One can view intentions as future paths that the agent *chooses to follow*. The intention-accessibility relation \mathcal{I} will be used to map the agent's current situation to all her intention-accessible worlds. We shall say that the agent intends a formula at a certain time if and only if it is true in all the agent's intention-accessible worlds of that time.

We saw above that the goal-accessible worlds of the agent can be viewed as the sub-worlds of the belief-accessible worlds in which the agent desires to be. Similarly, one can view intention-accessible worlds as sub-worlds of the goal-accessible worlds that the agent chooses to follow (i.e., to act upon). Thus, one moves from a belief-accessible world to a goal-accessible world by *desiring* future paths, and from a goal-accessible world to an intention-accessible world by *committing* to certain desired future paths.

The semantics for beliefs, goals, and intentions can be defined formally as follows:

$$\begin{aligned} M, v, w_t \models \text{BEL}(\phi) &\text{ iff } \forall w' \in \mathcal{B}_t^w M, v, w'_t \models \phi. \\ M, v, w_t \models \text{GOAL}(\phi) &\text{ iff } \forall w' \in \mathcal{G}_t^w M, v, w'_t \models \phi. \\ M, v, w_t \models \text{INTEND}(\phi) &\text{ iff } \forall w' \in \mathcal{I}_t^w M, v, w'_t \models \phi. \end{aligned}$$

We allow intentions over any well-formed formula, which means that one can have intentions about intentions, intentions about goals, intentions about beliefs, and intentions to do certain actions. Some might consider only the last type of intention to correspond with natural usage. While this is arguable, in our formalism the agent might have any type of intention but will only act on the last type of intention.

As an illustration of these ideas, Figure 3 shows one world b_1 that is belief-accessible from the current situation, say w_0 at t_0 , two worlds g_1 and g_2 that are goal-accessible from w_0 at t_0 , and two worlds i_1 and i_2 that are intention-accessible from w_0 at t_0 . It is clear from the figure that $i_1 \sqsubset g_1 \sqsubset b_1$ and $i_2 \sqsubset g_2$. One of the formulas that is true at t_0 in all the intended worlds is *succeeds*(d_1). Thus the agent intends *succeeds*(d_1). The sub-world relationship forces the agent to believe, as well as have the goal that *succeeds*(d_1). The agent intends to succeed and hence intends to carry out the action d_1 , but the agent cannot guarantee the ultimate success of her actions—that will be determined by the environment in which the agent is embedded. Thus, even though the above formula is true, it is not necessary, in the actual world, that the formula *succeeded*(d_1) be true.

From the figure it is clear that at t_0 , one of the goal formulas true in all goal accessible worlds is *inevitable*($\Diamond f$). This also implies that the agent believes that this goal is achievable; in other words, $\text{BEL}(\text{optional}(\Diamond f))$. From the beliefs, goals, and intentions of the agent, one can say that the agent believes that, if she succeeds in doing d_1 , she will achieve the goal f .

3.3 AXIOMATIZATION AND SEMANTIC CONDITIONS

So far, we have not provided any axioms or semantic conditions to capture the desired interrelationships among an agent's beliefs, goals, and intentions. We examine some of these below; additional constraints

are discussed elsewhere [14].

The axiomatization for beliefs is the standard weak-S5 (or KD45) modal system [11]. We adopt the D and K axioms for goals and intentions; i.e., goals and intentions have to be closed under implication and have to be consistent.

We also have the inference rule of necessitation [11] for beliefs, goals, and intentions. In other words, the agent believes all valid formulas, intends all valid formulas, and has them as a goal. Hence, like most possible-worlds formalisms, our logic also suffers from the logical omniscience problem [18]. This problem can be partly alleviated by adopting the *minimal-model* semantics of Chellas [4] and giving up the inference rule of necessitation and the K-axiom for beliefs, goals, and intentions. However, in this paper we adopt the more traditional modal-logic semantics.

Belief-Goal Compatibility:

The *Axiom of belief-goal compatibility* states that if the agent adopts an O-formula α as a goal, the agent believes that formula.

$$(AI1) \text{ GOAL}(\alpha) \supset \text{BEL}(\alpha).$$

The above axiom essentially states that, if the agent has the goal that *optional*(ψ) is true, she also believes it; i.e., there is at least one path in all the belief-accessible worlds in which ψ is true.

Consider, for example, the case where the formula ψ above is $\Diamond p$. The axiom then states that, if in all the goal-accessible worlds of the agent there is at least one path where eventually p becomes true, it must be the case that in all the belief-accessible worlds of the agent there is at least one path where eventually p is true. But note that, because of the branching nature of time, the agent need not believe she will ever reach the time point where p is true.

The notion of strong realism as described in Section 2 is captured by imposing the restriction that, for each and every belief-accessible world, there is a corresponding goal-accessible world such that the goal-world is a sub-world of the belief-world. This leads to the following semantic condition:

$$(CI1) \forall w' \in \mathcal{B}_t^w \exists w'' \in \mathcal{G}_t^w \text{ such that } w'' \sqsubseteq w'.$$

We shall use $\mathcal{B}_t^w \subseteq_{\text{super}} \mathcal{G}_t^w$ as a succinct notation for CI1. Such a relationship is shown in Figure 3, where $g_1 \sqsubset b_1$.

As both beliefs and goals are consistent, the relations \mathcal{B} and \mathcal{G} have to be *serial* (i.e., for any situation there is at least one belief-accessible world and at least one goal-accessible world). This ensures that, in the above semantic condition, we can find at least one belief-accessible world for which there is a goal-accessible

world.

To capture the notion of realism, Cohen and Levesque require, instead, that the goal relation \mathcal{G} be a subset of the belief relation \mathcal{B} ; i.e., $\mathcal{G} \subseteq \mathcal{B}$. As each possible world in their formalism is a time line, this imposes the condition that the chosen (or goal-accessible) worlds are compatible with the agent's belief-accessible worlds. In other words, $BEL(\phi) \supset GOAL(\phi)$ is an axiom in their formalism. This axiom forces the agent to adopt as goals certain inevitable facts about the world. As we shall see later, the different semantic condition used in our approach helps us avoid this problem of overcommitment.

Strong realism and realism are not the only ways of capturing the relationship between beliefs and goals. Elsewhere [16] we provide a different semantic relation between beliefs and goals that is suited to realizing other properties of these attitudes.

Goal-Intention Compatibility:

The *Axiom of goal-intention compatibility* states that, if the agent adopts an O-formula α as an intention, the agent should have adopted that formula as a goal to be achieved.

(AI2) $INTEND(\alpha) \supset GOAL(\alpha)$.

From the above axioms we have that, if the agent intends α , she believes in α as well. For example, if the agent intends to do an event e , she has the goal to (optionally) do e and also believes that she will (optionally) do e . Nested intentions lead to some interesting consequences. If the formula $INTEND(\text{inevitable}(\Diamond INTEND(\text{does}(e))))$ is true, then $BEL(\text{optional}(\Diamond INTEND(\text{does}(e))))$ is true and also $BEL(\text{optional}(\Diamond BEL(\text{does}(e))))$ is true.

Analogous to the semantic condition CI1 we have the semantic condition CI2, which imposes the restriction that for each and every goal-accessible world there is a corresponding intention-accessible world such that the intention-world is a sub-world of the goal-world.

(CI2) $\forall w' \in \mathcal{G}_t^w \exists w'' \in \mathcal{I}_t^w \text{ such that } w'' \sqsubseteq w'$.

We shall use $\mathcal{G}_t^w \subseteq_{\text{super}} \mathcal{I}_t^w$ as a succinct notation for CI2. Figure 3 illustrates the above semantic condition, where $i1 \sqsubset g1$ and $i2 \sqsubset g2$.

As discussed earlier, for each situation there is at least one goal-accessible world and at least one intention-accessible world.

Intentions leading to Actions:

The *Axiom of intention to action* (AI3) captures volitional commitment [2] by stating that the agent will act if she has an intention towards a single primitive

action e . Note that we have not said that the event e will occur successfully, just that the agent is committed to *trying* it. Whether the agent is successful or not depends on the environment in which she is embedded.

(AI3) $INTEND(\text{does}(e)) \supset \text{does}(e)$.

Thus, whenever an agent has an intention to do a particular primitive action, she will do that action. However, the axiom does not prevent the agent from doing actions that are not intended. Nor does it say anything about non-primitive actions or other forms of nested intentions.

Note that, if the agent has a *choice* of actions at the current time point, she would be incapable of acting *intentionally* until she deliberates and chooses one of them. One way of modeling this deliberation is to treat the process of deliberation itself as an action to be chosen by the agent [17]. An alternative approach would be to modify Axiom AI3 so that the agent arbitrarily chooses one of her intended actions and does that action.

Beliefs about Intentions:

If an agent has an intention, she believes that she has such an intention. The following axiom and semantic condition capture this notion.

(AI4) $INTEND(\phi) \supset BEL(INTEND(\phi))$.

(CI4) $\forall w' \in \mathcal{B}_t^w \text{ and } \forall w'' \in \mathcal{I}_t^w \text{ we have } w'' \in \mathcal{B}_t^{w'}$.

In Figure 3, this requires that $b1$ be \mathcal{I} -related to $i1$ and $i2$.

Beliefs about Goals:

If the agent has a goal to achieve ϕ , the agent believes that she has such a goal. This intuition can be captured by the following axiom and its corresponding semantic condition.

(AI5) $GOAL(\phi) \supset BEL(GOAL(\phi))$.

(CI5) $\forall w' \in \mathcal{B}_t^w \text{ and } \forall w'' \in \mathcal{G}_t^w \text{ we have } w'' \in \mathcal{B}_t^{w'}$.

In Figure 3, this requires that $b1$ be \mathcal{G} -related to $g1$ and $g2$.

Goals about Intentions:

If an agent intends to achieve ϕ , the agent must have the goal to intend ϕ . This requires the following axiom and semantic condition.

(AI6) $INTEND(\phi) \supset GOAL(INTEND(\phi))$.

(CI6) $\forall w' \in \mathcal{G}_t^w \text{ and } \forall w'' \in \mathcal{I}_t^w \text{ we have } w'' \in \mathcal{G}_t^{w'}$.

In Figure 3, this requires that $g1$ be \mathcal{I} -related to $i1$ and $i2$ and $g2$ be \mathcal{I} -related to $i1$ and $i2$.

One can strengthen Axioms AI4–AI6 by replacing each implications by an equivalence. This would result in $\text{INTEND}(\phi) \equiv \text{BEL}(\text{INTEND}(\phi)) \equiv \text{GOAL}(\text{INTEND}(\phi))$ and similarly $\text{GOAL}(\phi) \equiv \text{BEL}(\text{GOAL}(\phi))$, which has the effect of collapsing mixed, nested modalities to their simpler non-nested forms.

Awareness of Primitive Events

The next axiom requires the agent to be aware of all primitive events occurring in the world. Once again, we require only that the agent believe a primitive action has been done, not necessarily whether or not it was done successfully.

$$(\text{AI7}) \text{ done}(e) \supset \text{BEL}(\text{done}(e)).$$

No Infinite Deferral

Finally, we require the agent not to procrastinate with respect to her intentions. In other words, if an agent forms an intention, then some time in the future she will give up that intention. This axiom is similar to the one adopted by Cohen and Levesque [6], which requires that there be no infinite deferral of achievement goals.

$$(\text{AI8}) \text{ INTEND}(\phi) \supset \text{inevitable} \Diamond (\neg \text{INTEND}(\phi)).$$

The above axiom assumes that the intentions corresponding to maintenance goals are also dropped eventually. This could, if necessary, be avoided by restricting the formula ϕ in Axiom AI8 to be an action formula.

We shall refer to this set of eight axioms, AI1 – AI8, together with the standard axioms for beliefs and goals, as the *basic I-system*.

4 COMMITMENT AS AXIOMS OF CHANGE

So far we have treated intentions as a commitment to the performance of current actions. However, we have not formalized how these intentions guide or determine the agent's future commitment to her actions. In other words, we have not discussed how the agent's current intentions relate to her future intentions.

An alternative, proof-theoretic way of viewing the relationship between current and future intentions is as a process of intention maintenance and revision, or what we could intuitively think of as a commitment strategy. Different types of agent will have different commitment strategies. In what follows, we describe three different commitment strategies: *blind*, *single minded*, and *open minded*.

We define a *blindly committed agent* to be one who maintains her intentions until she *actually* believes

that she has achieved them. Formally, the axiom of blind commitment states that, if an agent intends that inevitably ϕ be eventually true, then the agent will inevitably maintain her intentions until she believes ϕ .

$$(\text{AI9a}) \text{ INTEND}(\text{inevitable} \Diamond \phi) \supset \text{inevitable}(\text{INTEND}(\text{inevitable} \Diamond \phi) \cup \text{BEL}(\phi)).$$

Depending on whether the formula ϕ is an event formula or not, we can capture commitment to actions (i.e., means) or to conditions that have to be true in the future (i.e., ends). Note also that the axiom is defined only for I-formulas (i.e., for intentions towards actions or conditions that are true of *all* paths in the agent's intention-accessible worlds); we do not say anything about the commitment of agents to *optionally* achieve particular means or ends.

A blind-commitment strategy is clearly very strong: the agent will eventually come to believe she has achieved her intentions or keep them forever. Relaxing this requirement, one can define *single-minded* commitment, in which the agent maintains her intentions as long as she believes that they are still options. More formally, we have the following axiom of single-minded commitment:

$$(\text{AI9b}) \text{ INTEND}(\text{inevitable} \Diamond \phi) \supset \text{inevitable}(\text{INTEND}(\text{inevitable} \Diamond \phi) \cup (\text{BEL}(\phi) \vee \neg \text{BEL}(\text{optional} \Diamond \phi))).$$

As long as she believes her intentions to be achievable, a single-minded agent will not drop her intentions and thus is committed to her goals. This requirement can also be relaxed. We define an *open-minded* agent to be one who maintains her intentions as long as these intentions are still her goals. In other words, the axiom of open-minded commitment can be stated as follows:

$$(\text{AI9c}) \text{ INTEND}(\text{inevitable} \Diamond \phi) \supset \text{inevitable}(\text{INTEND}(\text{inevitable} \Diamond \phi) \cup (\text{BEL}(\phi) \vee \neg \text{GOAL}(\text{optional} \Diamond \phi))).$$

We are now in a position to analyze the properties of different types of agent who adopt the basic I-system, together with one of the above axioms of commitment. Such an agent will be called a *basic agent*.

A basic agent blindly committed to her means (or ends) will inevitably eventually *believe* that she has achieved her means (or ends). This is because Axiom AI9a only allows future paths in which either the object of the intention is eventually believed or the intention is maintained forever. However, by Axiom AI8, the latter paths are not allowed, leading the agent to eventually believe that she has accomplished her intentions.

A basic single-minded agent reaches an identical conclusion only if she continues to believe, until the time she believes she has realized her intentions, that the

intended means (or ends) remains an option. Similarly, a basic open-minded agent will eventually believe she has achieved her intentions provided she maintains these intentions as goals until they are believed to have been achieved.

More formally, we have the following theorem for basic agents.

Theorem 1 :

(a) A basic, blindly committed agent, with the basic I-system and Axiom AI9a, satisfies the following property:

$$\text{INTEND}(\text{inevitable}(\Diamond\phi)) \supset \text{inevitable}(\Diamond\text{BEL}(\phi)).$$

(b) A basic single-minded agent, with the basic I-system and Axiom AI9b, satisfies the following property:

$$\begin{aligned} & \text{INTEND}(\text{inevitable}(\Diamond\phi)) \wedge \\ & \quad \text{inevitable}(\text{BEL}(\text{optional}(\Diamond\phi)) \cup \text{BEL}(\phi)) \\ & \supset \text{inevitable}(\Diamond\text{BEL}(\phi)). \end{aligned}$$

(c) A basic open-minded agent, with the basic I-system and Axiom AI9c, satisfies the following property:

$$\begin{aligned} & \text{INTEND}(\text{inevitable}(\Diamond\phi)) \wedge \\ & \quad \text{inevitable}(\text{GOAL}(\text{optional}(\Diamond\phi)) \cup \text{BEL}(\phi)) \\ & \supset \text{inevitable}(\Diamond\text{BEL}(\phi)). \end{aligned}$$

Proof:

(a) Assume the premise $\text{INTEND}(\text{inevitable}(\Diamond\phi))$. By Axiom AI9a we can conclude $\text{inevitable}(\text{INTEND}(\text{inevitable}(\Diamond\phi)) \cup \text{BEL}(\phi))$. By Axiom AI8 and the definition of weak until we can conclude $\text{inevitable}(\Diamond\text{BEL}(\phi))$. Cases (b) and (c) follow a similar line of reasoning. ♣

Consider now a competent agent [6] who satisfies the *Axiom of True Beliefs*, namely $\text{BEL}(\phi) \supset \phi$ (AI10). Under each of the different commitment strategies AI9a, AI9b, and AI9c, the competent agent will actually achieve her means (or ends), rather than just believe so. However, AI10 is often difficult for real agents to live up to, as it requires an agent to have true beliefs about the future realization of her intentions. By restricting Axiom AI10 to current beliefs only or to beliefs about primitive action formulas, we can define a less omniscient class of agents who will also inevitably eventually achieve their intentions.

Theorem 2 : Under the same conditions as Theorem 1, competent agents yield the conclusion $\text{inevitable}(\Diamond\phi)$ for all three types of commitment.

Proof: Follows directly from the proofs of Theorem 1 followed by the use of Axiom AI10. ♣

The above theorems, however, are not as useful as one would like. First, they do not make any use of Axiom

AI3. This means that the same result is achieved independent of whether or not the agent acts intentionally. Moreover, the second conjunct of the premises of (b) and (c) are conditions that have to be true in the real world and which are impossible for a situated agent to enforce; i.e., the agent cannot control these conditions. As a result, the above theorems, although interesting, do not provide a sufficient basis for a situated agent to reason about her intentions and actions.

Consider now an agent who always performs only intentional actions. This can be enforced by requiring the agent to intend a single primitive action at each and every time point. It is reasonable to expect that, in a world free of surprises, such an agent would maintain her beliefs after doing each intended action; i.e., she would not forget previously held beliefs.

More formally, we can state that an agent preserves a belief γ over an intentional action x if and only if (a) she intends to do x and (b) if she believes γ will hold after doing x , then after doing x , she does indeed believe γ :

$$\begin{aligned} & \text{INTEND}(\text{does}(x)) \wedge \\ & \quad (\text{BEL}(\text{optional} \bigcirc (\text{done}(x) \wedge \gamma)) \\ & \supset \text{optional} \bigcirc \text{BEL}(\text{done}(x) \wedge \gamma)). \end{aligned}$$

A single-minded agent who intends inevitably that ϕ is true in the future will inevitably come to believe ϕ provided that she carry out only intentional actions and that she preserve her beliefs about ϕ over these actions. If she were also competent, she would actually come to achieve ϕ .

Theorem 3 :

(a) A basic single-minded agent, with the basic I-system and Axiom AI9b, satisfies the following property:

$$\begin{aligned} & \text{INTEND}(\text{inevitable}(\Diamond\phi)) \wedge \\ & \quad \text{inevitable} \Box (\exists x (\text{INTEND}(\text{does}(x)) \wedge \\ & \quad (\text{BEL}(\text{optional} \bigcirc (\text{done}(x) \wedge (\Diamond\phi)))) \\ & \supset \text{optional} \bigcirc \text{BEL}(\text{done}(x) \wedge (\Diamond\phi))) \\ & \supset \text{inevitable}(\Diamond\text{BEL}(\phi)). \end{aligned}$$

(b) A competent single-minded agent, with the basic I-system, Axiom AI9b, Axiom AI10, and Axiom AI11 satisfies the following property:

$$\begin{aligned} & \text{INTEND}(\text{inevitable}(\Diamond\phi)) \wedge \\ & \quad \text{inevitable} \Box (\exists x (\text{INTEND}(\text{does}(x)) \wedge \\ & \quad (\text{BEL}(\text{optional} \bigcirc (\text{done}(x) \wedge (\Diamond\phi)))) \\ & \supset \text{optional} \bigcirc \text{BEL}(\text{done}(x) \wedge (\Diamond\phi))) \\ & \supset \text{inevitable}(\Diamond(\phi)). \end{aligned}$$

where the event variable x maps to a primitive event type.

Proof: (a) Assume the premise (i) $\text{INTEND}(\text{inevitable}(\Diamond\phi))$ and (ii) $\text{inevitable} \Box (\exists x (\text{INTEND}(\text{does}(x)) \wedge$

$$(\text{BEL}(\text{optional} \circ (\text{done}(x) \wedge (\Diamond\phi))) \supset \text{optional} \circ \text{BEL}(\text{done}(x) \wedge (\Diamond\phi))).$$

From (i) and Axioms AI2 and AI3 we have $\text{BEL}(\text{optional}(\Diamond\phi))$. From this conclusion and (ii), we have the conclusion $\text{inevitable}(\text{BEL}(\text{optional}(\Diamond\phi))) \cup \text{BEL}(\phi)$. Now we can use Theorem 1 to draw the desired conclusion.

Case (b) is identical to the above proof followed by the application of Axiom AI10. ♣

The second conjunct of the premises of both (a) and (b) can be weakened in several ways. First, Axiom AI3 allows us to drop the $\text{done}(x)$ formula in the real world. Second, the agent needs to act according to her intentions only until the moment that she achieves her intentions. In other words, $\text{inevitable}\Box$ can be replaced by the until operator.

Given that the agent will also believe the above theorem (by the inference rule of necessitation for beliefs) she will believe that, if she does only intentional actions and preserves her beliefs while doing so, she would ultimately achieve her goals. However, at the same time she can also reason that, if she is forced to do unintentional actions or does not maintain her beliefs, she may not be able to achieve her goals. Therefore, the "Little Nell" problem [12], in which an agent drops an intention precisely because he believes that its fulfillment will achieve his goal, does not arise.

Similar to the property of preservation of beliefs over intentional actions, one can introduce an analogous property of preservation of goals. This would allow open-minded agents to similarly achieve the object of their intentions.

We can also define other types of agent with mixed commitment strategies. For example, a particularly interesting commitment strategy is one in which the agent is open-minded with respect to ends but single-minded with respect to the means towards those ends. Such an agent is free to change the ends to which she aspires but, once committed to a means for realizing those ends, will not reconsider those means.

A *fanatically committed* agent [6] corresponds to a competent single-minded agent. Similarly, an agent with a *relativized commitment* is competent and open-minded with respect both to means and ends.

We are not suggesting that the above categorization is exhaustive or sufficient for describing realistic rational agents. Our aim in providing the above categorization has simply been to show that the formalism presented here provides a good basis for defining different types of agents and investigating their behavioral properties. It also lays the foundation for a more detailed analysis of reconsideration of intentions [2].

5 PROPERTIES OF THE LOGIC

There are two important aspects of belief-goal-intention interaction that have received attention in the literature [2, 5]. First, if an agent believes that a formula ϕ is *inevitably always* true (i.e., ϕ is true at all time points in all the future paths of all belief-accessible worlds), then the agent should *not* be forced to (a) adopt ϕ has a goal, or (b) intend ϕ . For example, given that "The earth is round" is true at all time points in all future paths of all belief-accessible worlds of the agent, the agent should not be forced to adopt this as a goal nor be forced to intend such inevitable facts. The same requirement holds for a slightly weaker form of belief; namely, the belief that a formula ϕ is *inevitably eventually* true (such as the belief about the rising of the sun). Clearly, an agent with such a belief should not be forced to adopt it as a goal nor intend it. Moreover, this requirement should hold no matter how persistent are the agent's beliefs. In particular, it should hold even if the agent *inevitably always* believes that a formula ϕ is *inevitably always* true.

Second, if an agent believes that a formula $\phi \supset \gamma$ is *inevitably always* true, and the agent intends ϕ (or has the goal ϕ), then the agent should *not* be forced to intend γ (or have the goal γ). In other words, an agent who intends to do a certain action should not be forced to intend all the *side-effects* of such an action. For example, an agent who intends to go to the dentist to have her tooth removed, but believes *inevitably* that going to the dentist will *always* cause her pain as a side-effect, should not be forced to intend herself pain [5]. As before, the above requirement also applies to the weaker form of belief and to persistent beliefs.

The above requirements are met by our formalism. While for every belief-accessible world there must be a goal-accessible world (and similarly for intentions), the converse need not hold. Thus, even if the agent believes that certain facts are inevitable, she is not forced to adopt them as goals (or as intentions). In this way, an agent believing that it is inevitable that pain always accompanies having a tooth filled, may yet have the goal (or intention) to have a tooth filled without also having the goal (or intention) to suffer pain. This relationship between belief, goal, and intention-accessible worlds is shown in Figure 3.

Let us define a binary relation $<_{\text{strong}}$ on the modal operators such that $\text{BEL} <_{\text{strong}} \text{GOAL} <_{\text{strong}} \text{INTEND}$. A modal formula $R_2(\phi)$ is said to be stronger than $R_1(\phi)$ if and only if $R_1 <_{\text{strong}} R_2$. We then have the following two propositions:

Proposition 1 : A modal formula does not imply a stronger modal formula. For example, if the agent believes (or *inevitably always* believes) that ϕ is true, she need not adopt ϕ as a goal. In other words, the follow-

ing formulas are satisfiable:

- (a) $\text{BEL}(\phi) \wedge \neg\text{GOAL}(\phi)$;
- (b) $\text{inevitable}(\Box\text{BEL}(\phi)) \wedge \neg\text{GOAL}(\phi)$.

General case: In general, the above results hold if BEL is substituted by R_1 and GOAL by R_2 , where $R_1 <_{\text{strong}} R_2$.

Proof: For every belief-accessible world there has to be a goal-accessible world. But the goal relation \mathcal{G} can map to worlds that do not correspond to any belief accessible world. Thus, if in one such world the formula ϕ is not true, the agent will not have ϕ as a goal. This shows the satisfiability of Case (a). The satisfiability of Case (b) follows a similar pattern. ♣

Proposition 2 : A modal operator is not closed under implication with respect to a weaker modality. For example, the following formulas are satisfiable:

- (a) $\text{GOAL}(\phi) \wedge \text{BEL}(\text{inevitable}(\Box(\phi \supset \gamma))) \wedge \neg\text{GOAL}(\gamma)$;
- (b) $\text{GOAL}(\phi) \wedge \text{inevitable}(\Box\text{BEL}(\text{inevitable}(\Box(\phi \supset \gamma)))) \wedge \neg\text{GOAL}(\gamma)$.

General case: In general, the above results hold if BEL is substituted by R_1 and GOAL by R_2 , where $R_1 <_{\text{strong}} R_2$.

Proof: For every belief-accessible world there has to be a goal-accessible world. But the goal relation \mathcal{G} can map to worlds that do not correspond to any belief-accessible world. Thus, if in one such world the formula $\phi \supset \gamma$ is not true, the agent will not have γ as a goal. This shows the satisfiability of Case (a). The satisfiability of Case (b) follows a similar pattern. ♣

Both the above propositions deal with the stronger form of beliefs; namely, that the agent believes *inevitably always* ϕ . They can be suitably modified for the weaker form as well.

Note that although we have the above propositions, the agent's goals and intentions are closed under implication. In other words, the following formulas are valid formulas in our system:

$$\begin{aligned} \text{INTEND}(\phi) \wedge \text{INTEND}(\phi \supset \gamma) &\supset \text{INTEND}(\gamma). \\ \text{GOAL}(\phi) \wedge \text{GOAL}(\phi \supset \gamma) &\supset \text{GOAL}(\gamma). \end{aligned}$$

Moreover, although an agent need not have as a goal or intention such inevitable facts, this does not prevent her from reasoning about them. Thus, for example, on adopting the intention to go to the dentist, the agent could still use her beliefs about the certainty of accompanying pain in deciding to take along a strong analgesic.

Cohen and Levesque define a notion of persistent goal which appears to have some of the above properties. However, these properties are obtained by appealing to the temporal nature of persistent goals, rather than any intrinsic properties of beliefs, goals, or intentions [1]. For example, Cohen and Levesque can only avoid intending the side effects of any intended action if, at some time point in the future, the agent does not believe that the side effect will result from performance of the intended action. The problem remains, however, for cases in which the agent, for example, always believes that the side effect will occur. In other words, Case (b) is not satisfiable for the above propositions for the non-trivial cases. In contrast, in our formalism the agent is not forced to adopt unwanted goals or intentions on account of her beliefs, no matter how strong or persistent these beliefs are.

6 COMPARISON AND CONCLUSION

Bratman [2] argues against the reducibility of intentions to an agent's beliefs and desires and treats intentions as partial plans of action to which the agent is committed. He then goes on to show how the agent's existing beliefs, desires, and intentions form a background for future deliberation. Following this line of argument, we have introduced a logical formalism which accords a primary status to intentions. Further, by adopting certain axioms of change, we have shown how the present beliefs, goals, and intentions of an agent constrain her future attitudes.

Philosophically, our approach differs from that of Cohen and Levesque [6] in that it treats intention as a basic attitude and shifts the emphasis of future commitment from the definition of intention to the process of intention revision. Semantically, our approach differs in that we distinguish between the choice available to the agent in choosing her actions and her beliefs about which worlds are possible. In addition, we specify an interrelationship between beliefs, goals, and intentions that allows us to avoid all variations of the problem of unwanted side effects.

We have only considered constraints on the maintenance of intentions. Important aspects of rational behavior that concerns intention formation by *deliberation* and intention modification in the light of changing circumstances or *reconsideration* [2] have not been dealt with in this paper. These are separate topics which will be considered elsewhere; the sub-world relationship between beliefs, goals, and intentions provides useful techniques for analyzing these issues.

In summary, we have presented a theory of intention that treats intentions on a par with the agent's beliefs and goals. By introducing various axioms of change, we were able to categorize a variety of rational agents

and their commitment strategies. We also captured, for the first time, the process of belief, goal, and intention revision, which is crucial for understanding rational behavior [1]. Although there are many aspects of a theory of rational agency that we have not addressed, we believe that we have presented a formalism that provides a foundation upon which such a theory can be constructed.

Acknowledgements

The authors would like to thank Phil Cohen, Martha Pollack, Liz Sonenberg, David Israel, Kurt Konolige, Douglas Appelt, and Félix Ingrand for valuable discussions and comments on the contents of this paper.

References

- [1] J. Allen. Two views of intention: Comments on Bratman and on Cohen and Levesque. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, Cambridge, Ma., 1990.
- [2] M. E. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, Massachusetts, 1987.
- [3] M. E. Bratman, D. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.
- [4] B. F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.
- [5] P. R. Cohen and H. J. Levesque. Persistence, intention and commitment. In M. P. Georgeff and A. L. Lansky, editors, *Proceedings of the 1986 workshop on Reasoning about Actions and Plans*, pages 297–340. Morgan Kaufmann Publishers, San Mateo, CA, 1987.
- [6] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.
- [7] E. A. Emerson and J. Srinivasan. Branching time temporal logic. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, pages 123–172. Springer-Verlag, Berlin, 1989.
- [8] P. Gardenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Bradford Book, MIT Press, Cambridge, MA., 1988.
- [9] M.P. Georgeff and F.F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Detroit, MI, 1989.
- [10] J. Y. Halpern and Y. O. Moses. A guide to the modal logics of knowledge and belief. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, Los Angeles, CA, 1985.
- [11] G. E. Hughes and M. J. Cresswell. *A Companion to Modal Logic*. Methuen & Co. Ltd., London, England, 1984.
- [12] D. V. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.
- [13] A. S. Rao and N. Y. Foo. Minimal change and maximal coherence: A basis for belief revision and reasoning about actions. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-89*, Detroit, MI, 1989.
- [14] A. S. Rao and M. P. Georgeff. Deliberation and the formation of intentions. Technical Report 10, Australian AI Institute, Carlton, Australia, 1990.
- [15] A. S. Rao and M. P. Georgeff. A formal model of intentions. In *Pacific Rim International Conference on Artificial Intelligence, PRICAI-90*, Nagoya, Japan, November 1990.
- [16] A. S. Rao and M. P. Georgeff. Asymmetry thesis and side-effect problems in linear time and branching time intention logics. Technical Report 13, Australian AI Institute, Carlton, Australia, 1991.
- [17] S. Russell and E. Wefald. Principles of metareasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, 1989.
- [18] M. Y. Vardi. On epistemic logic and logical omniscience. In J. Y. Halpern, editor, *Proceedings of the First Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 293–306, San Mateo, California, 1986. Morgan Kaufmann Publishers.

A Nonmonotonic Assumption-Based TMS Using Stable Bases

William L. Rodi
 MIT Earth Resources Lab
 42 Carleton Street
 Cambridge, MA 02139

Stephen G. Pimentel
 Adroit Systems Inc.
 809 North Royal Street
 Alexandria, VA 22314

Abstract

The stable model semantics of [Gelfond & Lifschitz 88] has become a standard method of supplying semantics for nonmonotonic logic programs, and has been applied to truth maintenance systems as well. In this paper, we introduce a new construct called *stable bases* which are in some sense the dual of stable models and from which stable models can be easily derived. Stable bases are distinguished by the fact that they are contained within a (potentially small) subset of the proposition set¹ and are thus easier to compute than stable models. We apply stable bases in the design of a truth maintenance system, the NATMS, which integrates nonmonotonic and abductive reasoning in a fully general fashion. The NATMS extends the ATMS of [de Kleer 86] by labeling each proposition with both the assumption sets and stable bases which together allow the proposition to be derived. The NATMS thus accounts for the multiple stable models which may result from nonmonotonicity in the justification set, yet defaults to the behavior of the ATMS in the presence of purely monotonic justifications.

1 INTRODUCTION

The feature of the truth maintenance problem which makes it computationally difficult is the necessity of dealing with multiple “canonical” models corresponding to a given justification set. This multiplicity stems from two orthogonal sources. First, a system may have multiple *extensions*, which arise from nonmonotonicity in the justifications and are strictly analogous to the multiple extensions of nonmonotonic logic. Second, a system may have multiple *contexts*, which arise from

distinct assumption sets and are formalizable wholly within the framework of classical propositional logic [Reiter & de Kleer 87]. Traditional truth maintenance systems [Doyle 79; de Kleer 86], as well as more recent ones [Pimentel & Cuadrado 89; Junker 89; Eshghi 90], have dealt with at most *one* of these phenomena. The system of [Dressler 88] is the first TMS which correctly deals with both.

This paper describes a new TMS which also deals simultaneously with the two types of multiplicity. Like the ATMS of [de Kleer 86] it is an assumption-based system which maintains contexts for multiple sets of assumptions. However, our system extends the ATMS in that it accepts nonmonotonic justifications and, for each set of assumptions it maintains, finds the multiple extensions which may arise from this nonmonotonicity. Our system defaults to the behavior of the ATMS when given purely monotonic justifications, a property we propose as a simple and intuitive requirement for any system which integrates nonmonotonicity into the ATMS. The system of [Dressler 88] also satisfies this requirement, but other attempts to extend the ATMS to handle nonmonotonicity, such as [Junker 89] and [Eshghi 90], have not.

We formulate our TMS within the logic of Horn clauses generalized by negation as failure. The semantics of the system is then precisely defined in terms of *stable models*. Stable models were introduced in [Gelfond & Lifschitz 88] to provide a declarative semantics for logic programs with negation and are closely related to the “stable expansions” of autoepistemic logic [Moore 85] and the “finitely grounded extensions” of [Doyle 83]. As a result, the behavior of our TMS can be immediately related to the use of negation as failure in logic programs and to other nonmonotonic formalisms proposed in the literature.

The algorithm we present for our TMS employs a new construct called a *stable base*. We show that each stable model of a logic program corresponds to a unique stable base, and that stable models and bases are simply related to one another. Stable bases assume

¹or Herbrand base, in the case of first-order theories.

the role of the “out-assumptions” in the system of [Dressler 88] and are related to the “extension bases” of [Junker 89] and “stable generators” of [Eshghi 90]. Stable bases have the advantage that they render unnecessary the introduction of new propositions and justifications. Further, they help to elucidate the interaction between multiple extensions and assumptions.

The remainder of this paper is organized as follows. In Section 2, we review the stable model semantics of generalized Horn clauses and introduce stable bases. In Section 3, we use this logic to give a declarative specification of the new nonmonotonic ATMS along with the algorithms for label propagation and querying. Section 4 discusses the implementation of the new TMS. Section 5 gives a detailed comparison of our system with the work of other authors who have proposed systems of a similar nature. Finally, Section 6 states our conclusions.

2 THE LOGIC OF GENERALIZED HORN CLAUSES

2.1 LANGUAGE

We let P denote a set of proposition symbols. We will take as our language the set of generalized Horn clauses, that is, the set of formulas of the form

$$c \leftarrow a_1, \dots, a_n, \neg b_1, \dots, \neg b_m, \quad m, n \geq 0$$

where $c, a_1, \dots, a_n, b_1, \dots, b_m \in P$. If $m = 0$, the clause is a (non-generalized) Horn clause.

We will denote clauses by α and sets of clauses by Σ . We will sometimes treat a subset of propositions $A \subseteq P$ as a set of clauses with the understanding that $p \in A$ corresponds to $p \leftarrow$. Given a clause α , we write $\text{horn}(\alpha)$ for the clause resulting from removing all negative literals from α , $\text{head}(\alpha)$ for the head of α , $\text{Pos}(\alpha)$ for the set of propositions appearing as positive literals in the body of α , and $\text{Neg}(\alpha)$ for the set of propositions which are negated in α .

An interpretation I is a subset of P . $I \subseteq P$ is a *model* of Σ if I satisfies each clause in Σ according to the standard definition of satisfaction [Lloyd 87]. If Σ consists only of Horn clauses, then it has a unique minimal model [Lloyd 87], which we will denote $\text{Min}(\Sigma)$.

2.2 STABLE MODELS

If Σ consists of generalized Horn clauses, on the other hand, it may have multiple minimal models, many of which will be unintuitive with respect to the “intended” semantics of the clauses. [Gelfond & Lifschitz 88] introduced the more restrictive class of stable models to alleviate this difficulty. To define the stable models, [Gelfond & Lifschitz 88] introduce a certain transform on Σ with respect to an interpretation I :

$$\Sigma_I = \{ \text{horn}(\alpha) \mid \alpha \in \Sigma, I \cap \text{Neg}(\alpha) = \emptyset \}. \quad (1)$$

I is said to be a *stable interpretation* of Σ if $I = \text{Min}(\Sigma_I)$.

It is shown in [Gelfond & Lifschitz 88] that every stable interpretation of Σ is in fact a minimal model of Σ (although not all minimal models are stable). Therefore, we may simply speak of the stable interpretations of Σ as its *stable models*.

Recall the notion of SLD-refutation [Lloyd 87] from Horn clauses. For a set of Horn clauses Σ , we write $\Sigma \vdash p$ if there is an SLD-refutation of p from Σ . We then have that an interpretation I is a stable model of Σ if and only if it satisfies

$$\text{for all } p \in P: p \in I \text{ iff } \Sigma_I \vdash p. \quad (2)$$

This follows immediately from the completeness and soundness of SLD-refutation [Lloyd 87].

2.3 STABLE BASES

Our TMS algorithm is based on the realization that, in order to compute the stable models of Σ , it is not necessary to check the condition in (2) for every proposition. Rather, it suffices to test only a (potentially small) subset of P , namely those propositions which occur in negative literals in Σ . Let N denote this set of propositions:

$$N = \bigcup_{\alpha \in \Sigma} \text{Neg}(\alpha). \quad (3)$$

Definition 2.1 (stable base) $B \subseteq N$ is a *stable base* of Σ if it satisfies,²

$$\text{for all } p \in N: p \in \overline{B} \text{ iff } \Sigma_{\overline{B}} \vdash p. \quad (4)$$

We can interpret (4) as saying that \overline{B} is stable over the subset of propositions in N . In comparison, (2) states that I is stable over the full set of propositions.

Let us now examine the relationship between stable bases and stable models. We start by showing the following lemma.

Lemma 2.1 For any $I, I' \subseteq P$, if $I \cap N = I' \cap N$ then $\Sigma_I = \Sigma_{I'}$.

Proof: Assume $I \cap N = I' \cap N$. Then for any clause $\alpha \in \Sigma$ we must have $\text{Neg}(\alpha) \cap I = \text{Neg}(\alpha) \cap I'$ since $\text{Neg}(\alpha) \subseteq N$. Thus, (1) implies $\Sigma_I = \Sigma_{I'}$. ■

This lemma allows us to prove the following theorem which establishes a one-to-one correspondence between stable models and stable bases.

Theorem 2.1 I is a stable model of Σ and $B = \overline{I} \cap N$ iff B is a stable base of Σ and $I = \text{Min}(\Sigma_{\overline{B}})$.

² $\overline{B} = P - B$, the complement of B in P .

Proof: (\Rightarrow) Let I be a stable model of Σ and set $B = \bar{I} \cap N$. Clearly,

$$\bar{B} = I \cup \bar{N} \quad (5)$$

$$\bar{B} \cap N = I \cap N. \quad (6)$$

Lemma 2.1 and (6) imply $\Sigma_{\bar{B}} = \Sigma_I$.

To show that B is a stable base of Σ we must show $p \in \bar{B}$ iff $\Sigma_{\bar{B}} \vdash p$ for all $p \in N$. Let p be any member of N . From (5) we see that $p \in \bar{B}$ iff $p \in I$ and, since I is a stable model, iff $\Sigma_I \vdash p$. Since $\Sigma_I = \Sigma_{\bar{B}}$, we have $p \in \bar{B}$ iff $\Sigma_{\bar{B}} \vdash p$.

Since I is a stable model, then $I = \text{Min}(\Sigma_I)$ and, hence, $I = \text{Min}(\Sigma_{\bar{B}})$.

(\Leftarrow) Let B be a stable base of Σ and set $I = \text{Min}(\Sigma_{\bar{B}})$. Clearly, $I = \{p \mid \Sigma_{\bar{B}} \vdash p\}$, which implies

$$I \cap N = \{p \in N \mid \Sigma_{\bar{B}} \vdash p\} \quad (7)$$

$$\bar{I} \cap N = \{p \in N \mid \Sigma_{\bar{B}} \not\vdash p\}. \quad (8)$$

Since B is a stable base, (7) becomes

$$I \cap N = \{p \in N \mid p \in \bar{B}\} = \bar{B} \cap N$$

Therefore, $\Sigma_I = \Sigma_{\bar{B}}$ (Lemma 2.1) and hence $I = \text{Min}(\Sigma_I)$, implying I is a stable model.

Similarly, (8) becomes

$$\bar{I} \cap N = \{p \in N \mid p \notin \bar{B}\} = B \cap N.$$

Since $B \subseteq N$, we obtain $\bar{I} \cap N = B$. ■

We note the special case of this theorem when Σ contains only Horn clauses. We then have $N = \emptyset$ and $B = \emptyset$ as the only stable base of Σ . Theorem 2.1 reduces to: I is a stable model of Σ iff $I = \text{Min}(\Sigma)$.

More generally, Theorem 2.1 implies that the set of stable models of Σ and the set of its stable bases can be related by the bijective mappings

$$I = \text{Min}(\Sigma_{\bar{B}})$$

$$B = \bar{I} \cap N.$$

Example 1: Let $P = \{a, b, c\}$ and let Σ contain the two clauses:

$$\begin{aligned} a &\leftarrow \neg b \\ b &\leftarrow \neg c. \end{aligned}$$

We then have $N = \{b, c\}$ and Σ has the solitary stable base $\{c\}$ and stable model $\{b\}$.

Example 2: Again let $P = \{a, b, c\}$ but now let Σ contain the three clauses:

$$\begin{aligned} a &\leftarrow \neg b \\ b &\leftarrow \neg c \\ c &\leftarrow a. \end{aligned}$$

As in the last example, $N = \{b, c\}$. Σ now has two stable bases: $\{b\}$ and $\{c\}$. The stable models associated with these stable bases are, respectively, $\{a, c\}$ and $\{b\}$.

3 THE NONMONOTONIC ATMS

In this section, we will describe a truth maintenance system, which we call the NATMS, which handles both nonmonotonicity and assumption-based reasoning. Our definition of the NATMS is motivated by a simple and intuitive requirement: in the case of purely monotonic justifications the system should behave exactly as a monotonic ATMS, i.e., the system of [de Kleer 86]. Most previous efforts in this area have not satisfied this requirement, as we show in Section 5.

Definition 3.1 (Nonmonotonic ATMS)

A nonmonotonic ATMS (NATMS) is a quadruple $(P, A_{\max}, p_{\perp}, \Sigma)$ where

1. P is a finite set of propositions,
2. $A_{\max} \subseteq P$ is a subset of propositions which may be used as "assumptions,"
3. $p_{\perp} \in P$ is a proposition which represents "contradiction," and
4. Σ is a finite set of generalized Horn clauses defined over a (possibly proper) subset of P .

An NATMS such that Σ contains only Horn clauses defines the ATMS of [de Kleer 86].

Note that $(P, A_{\max}, p_{\perp}, \Sigma)$ defines a "snapshot" of the NATMS at a given point in time. In practice, P , A_{\max} and Σ may be augmented incrementally as a problem solver interacts with the system. Our main concern here is a declarative specification of a snapshot NATMS.

The function of the NATMS is to determine, for each $p \in P$, every set of assumptions $A \subseteq A_{\max}$ such that p can be derived, and p_{\perp} cannot be derived, from $\Sigma \cup A$. To define this precisely, let us consider first the ATMS. If Σ is restricted to Horn clauses, then for any A , $\Sigma \cup A$ will have a unique stable model, $I = \text{Min}(\Sigma \cup A)$, and we will have $p \in I$ iff $\Sigma \cup A \vdash p$. The function of the ATMS can then be said to compute for each $p \in P$ the set $\text{QUERY}(p)$ defined by

$$\text{QUERY}(p) = \{A \subseteq A_{\max} \mid \Sigma \cup A \vdash p, \Sigma \cup A \not\vdash p_{\perp}\}. \quad (9)$$

$\text{QUERY}(p)$ (actually its minimal elements) is what de Kleer's system returns in answer to a query about p .

If, on the other hand, Σ contains generalized Horn clauses, then $\Sigma \cup A$ is not guaranteed to have a unique stable model but may have zero, one or more stable models. This requires that we in some way parameterize assumption sets with the stable models in which a given proposition p can be derived. It is convenient to use stable bases for this purpose. That is, we will extend the definition of $\text{QUERY}(p)$ to include pairs (A, B)

where A is an assumption set and B is a stable base of $\Sigma \cup A$. To state this precisely, we first define the set

$$\text{ALL} = \{(A, B) \mid A \subseteq A_{\max}, B \subseteq N, A \cap B = \emptyset\} \quad (10)$$

where N is given by (3). The condition $A \cap B = \emptyset$ recognizes that a stable model of $\Sigma \cup A$ must include A itself, implying its stable base must be disjoint from A . Thus, of all the possible pairs $(A, B) \in 2^{A_{\max}} \times 2^N$, ALL excludes those in which B is not potentially a stable base of $\Sigma \cup A$. We extend the definition of QUERY to

$$\begin{aligned} \text{QUERY}(p) = & \{(A, B) \in \text{ALL} \mid \\ & \Sigma_{\overline{B}} \cup A \vdash p, \Sigma_{\overline{B}} \cup A \not\vdash p_{\perp}, \\ & B \text{ is a stable base of } \Sigma \cup A\}. \end{aligned} \quad (11)$$

The chief function of the NATMS is to compute $\text{QUERY}(p)$ as defined in (11).

Observe that $\Sigma_{\overline{B}} \cup A = (\Sigma \cup A)_{\overline{B}}$, since $A_{\overline{B}} = A$, and that $\Sigma_{\overline{B}} \cup A \vdash p$ iff $p \in \text{Min}(\Sigma_{\overline{B}} \cup A)$. Further, Theorem 2.1 guarantees $\text{Min}(\Sigma_{\overline{B}} \cup A)$ is a stable model iff B is a stable base of $\Sigma \cup A$. Therefore, $\text{QUERY}(p)$ maps one-to-one onto the set of (A, I) such that $p \in I$, $p_{\perp} \notin I$, and I is a stable model of $\Sigma \cup A$. For each $(A, B) \in \text{QUERY}(p)$, we can think of A as a set of “in-assumptions” and B as a set of “out-assumptions” which together allow p to be derived from Σ .

Considering again the monotonic case, we have $N = \emptyset$, $\Sigma_{\overline{B}} = \Sigma$ for any B , and $B = \emptyset$ as the unique stable base of $\Sigma \cup A$ for any A . As a result, $\text{QUERY}(p)$ in (11) reduces to (9) but with each A paired with an empty set of “out-assumptions.”

3.1 COMPUTING QUERIES

The computation of $\text{QUERY}(p)$ is divided into two stages. The first is a fully incremental label-propagation algorithm. Like the ATMS, the NATMS maintains a *label* for each proposition which is assigned an initial value when the proposition is introduced into the TMS. Labels are updated each time a new clause is appended to Σ .

The second stage is a non-incremental constraint satisfaction procedure performed upon labels of the propositions in N in order to compute the stable bases. The cost of the procedure reduces to zero when $N = \emptyset$.

Both stages of the computation rely upon the ability to perform set-theoretic operations (chiefly intersection, union, set difference, and subset test) upon subsets of ALL, defined in (10). The implementation of these operations is discussed in Section 4.

Looking at the definition of $\text{QUERY}(p)$ in (11), we note that $\text{QUERY}(p)$ can be decomposed as

$$\text{QUERY}(p) = \text{BASE} \cap \text{LABEL}(p) - \text{LABEL}(p_{\perp}) \quad (12)$$

where

$$\text{LABEL}(p) = \{(A, B) \in \text{ALL} \mid \Sigma_{\overline{B}} \cup A \vdash p\} \quad (13)$$

and

$$\begin{aligned} \text{BASE} = & \{(A, B) \in \text{ALL} \mid \\ & B \text{ is a stable base of } \Sigma \cup A\}. \end{aligned} \quad (14)$$

As its name suggests, $\text{LABEL}(p)$ is the label we will associate with p , while BASE gives us the stable bases for the various assumption sets. Below we will show how $\text{LABEL}(p)$ and BASE can be computed. Given these sets, we can compute $\text{QUERY}(p)$ directly using (12).

Again, it is interesting to consider formulas (12) and (14) in the strictly monotonic case. When Σ contains only Horn clauses we have $N = \emptyset$, which is a stable base of $\Sigma \cup A$ for any A . Hence, (14) reduces to $\text{BASE} = \text{ALL} = 2^{A_{\max}} \times \{\emptyset\}$. From (12) and (13) this yields

$$\begin{aligned} \text{QUERY}(p) = & \text{LABEL}(p) - \text{LABEL}(p_{\perp}) \\ = & \{(A, \emptyset) \mid \Sigma \cup A \vdash p, \Sigma \cup A \not\vdash p_{\perp}\} \end{aligned}$$

confirming our earlier observation. Thus, we see that the computation of BASE is required only by the presence of nonmonotonicity in Σ .

3.2 INCREMENTAL LABEL MAINTENANCE

In the following discussion, we will disregard the obvious bookkeeping functions needed for updating P , N and Σ as propositions and clauses are introduced to the system. We consider only the updating of the proposition labels.

If a proposition p does not occur in Σ , then it will be present in a stable model of $\Sigma \cup A$ only if $p \in A$. Therefore, when p is first introduced into the NATMS, we set $\text{LABEL}(p) := \text{INASM}(p)$, where $\text{INASM}(p)$ is defined by

$$\text{INASM}(p) = \{(A, B) \in \text{ALL} \mid p \in A\}. \quad (15)$$

Note that if $p \notin A_{\max}$, we obtain $\text{INASM}(p) = \emptyset$ owing to the exclusion of $A \not\subseteq A_{\max}$ from ALL.

After a clause α has been appended to Σ , the labels are updated by the label-propagation algorithm update-label(α). This algorithm, described below, is very similar to that used in de Kleer's system.

update-label (α)

```

More := { $\alpha$ };
while More ≠ ∅ do
  choose  $\beta \in \text{More}$ ;
  More := More - { $\beta$ };
  OUT := {(A, B) ∈ ALL | B ⊇ \text{Neg}( $\beta$ )};
  NEW := OUT ∩  $\bigcap_{p \in \text{Pos}(\beta)} \text{LABEL}(p)$ ;
  if NEW ⊈ \text{LABEL}(\text{head}( $\beta$ )) then
    \text{LABEL}(\text{head}( $\beta$ )) := \text{LABEL}(\text{head}( $\beta$ )) ∪ NEW;
  More := More ∪ { $\gamma \in \Sigma | \text{head}(\beta) \in \text{Pos}(\gamma)$ };
```

We can see from update-label that the addition of a clause to Σ induces the addition of elements to $\text{LABEL}(p)$. Thus, $\text{LABEL}(p)$ is a monotone increasing (with respect to set inclusion) function of Σ .

3.3 COMPUTING STABLE BASES

Given the algorithm for incrementally computing $\text{LABEL}(p)$, our next step is to find a method for computing BASE as defined in (14). We begin by defining the set

$$\text{OUTASM}(p) = \{(A, B) \in \text{ALL} \mid p \in B\}. \quad (16)$$

If we examine Definition 2.1, it turns out that BASE can be expressed in terms of $\text{LABEL}(p)$ and $\text{OUTASM}(p)$ for $p \in N$. Specifically, we can carry out a boolean expansion of (4) within (14) to obtain

$$\text{BASE} = \bigcap_{p \in N} \text{LABEL}(p) \Delta \text{OUTASM}(p) \quad (17)$$

where Δ denotes the symmetric difference between sets. From Definition 2.1, it should be obvious that $(A, B) \in \text{BASE}$ just in case B is a stable base of $\Sigma \cup A$. Note that the intersection (17) is computed over N rather than P . If Σ is monotonic, then $N = \emptyset$ and BASE defaults to ALL without computation.

While $\text{LABEL}(p)$ is monotone with respect to Σ , and $\text{OUTASM}(p)$ is a constant set, the presence of the symmetric difference in (17) implies that BASE is not a monotone function of Σ . Because of this, it appears that the above formula for BASE must be computed “from scratch” after each change to the label of a negated proposition. In the next subsection, we consider an alternative formula for BASE which allows part of the computation to be done incrementally.

3.4 MINIMAL SET REPRESENTATIONS

We can factor BASE into a difference between sets as follows. For any sets X and Y we have $X \Delta Y = (X \cup Y) - (X \cap Y)$. Further,

$$\bigcap_i (X_i - Y_i) = \bigcap_i X_i - \bigcup_i Y_i$$

for any sets $X_1, Y_1, X_2, Y_2, \dots$. Applying these identities to (17) allows us to write

$$\text{BASE} = \text{GOOD} - \text{BAD} \quad (18)$$

where

$$\text{GOOD} = \bigcap_{p \in N} \text{LABEL}(p) \cup \text{OUTASM}(p) \quad (19)$$

$$\text{BAD} = \bigcup_{p \in N} \text{LABEL}(p) \cap \text{OUTASM}(p). \quad (20)$$

It is informative to translate the formulas for GOOD and BAD back into conditions on A and B . We have $(A, B) \in \text{GOOD}$ iff

$$\text{for all } p \in N: p \in \overline{B} \text{ implies } \Sigma_{\overline{B}} \cup A \vdash p.$$

$(A, B) \notin \text{BAD}$ iff

$$\text{for all } p \in N: \Sigma_{\overline{B}} \cup A \vdash p \text{ implies } p \in \overline{B}.$$

The conjunction of these two conditions yields (4).

Using (18) in (12) obtains

$$\text{QUERY}(p) = \text{LABEL}(p) \cap \text{GOOD} - \text{NOGOOD} \quad (21)$$

where

$$\text{NOGOOD} = \text{LABEL}(p_{\perp}) \cup \text{BAD}. \quad (22)$$

(21)–(22) state that $\text{QUERY}(p)$ comprises the elements of $\text{LABEL}(p) \cap \text{GOOD}$ but with those in $\text{LABEL}(p_{\perp}) \cup \text{BAD}$ removed. Compared to the ATMS, where $\text{QUERY}(p) = \text{LABEL}(p) - \text{LABEL}(p_{\perp})$, we see that GOOD acts as a filter on $\text{LABEL}(p)$ while BAD augments $\text{LABEL}(p_{\perp})$. The elements of $\text{LABEL}(p_{\perp})$ are known as “nogoods” in the terminology of [de Kleer 86] and it is clear that the elements of BAD play a similar role in the NATMS. Hence our use of the notation NOGOOD .

We note that both GOOD and NOGOOD are monotone functions of the proposition labels, which in turn are monotone functions of Σ . NOGOOD , as a union of labels, can thus be computed incrementally, whereas the full intersection yielding GOOD must be recomputed whenever the labels of the propositions in N have changed.

The real value of this factoring of BASE and $\text{QUERY}(p)$ stems from the fact that $\text{LABEL}(p)$ and $\text{OUTASM}(p)$ can be represented purely by their minimal elements. Let us define a partial order \preceq on ALL such that

$$(A, B) \preceq (A', B') \text{ iff } A \subseteq A', B \subseteq B'.$$

Given any $X \subseteq \text{ALL}$, we define $\mu X \subseteq X$ to contain the minimal elements of X with respect to \preceq :

$$\begin{aligned} \mu X &= \{(A, B) \in X \mid \\ &\quad \text{for all } (A', B') \in X: \\ &\quad (A', B') \preceq (A, B) \text{ implies } (A', B') = (A, B)\}. \end{aligned}$$

We will say that X is *upwardly closed* with respect to \preceq if membership in X is monotonic with respect to \preceq , i.e., $(A, B) \in X$ and $(A, B) \preceq (A', B')$ imply $(A', B') \in X$. If X is upwardly closed in this sense it is totally determined by μX :

$$\begin{aligned} X &= \{(A, B) \in \text{ALL} \mid \\ &\quad \text{there exists } (A', B') \in \mu X: (A', B') \preceq (A, B)\}. \end{aligned}$$

It is obvious from (16) that $\text{OUTASM}(p)$ is upwardly closed with respect to \preceq . $\text{LABEL}(p)$ is also upwardly closed owing to the facts that $\Sigma_{\overline{B}} \cup A$ is a monotonic function of both A and B , and derivability is monotonic with respect to a set of Horn clauses. It follows from (19), (20) and (22) that GOOD and NOGOOD are upwardly closed with respect to \preceq since these sets are obtained via unions and intersections of $\text{LABEL}(p)$ and

$\text{OUTASM}(p)$. Therefore, in complete analogy with the ATMS, the NATMS may return, in place of $\text{QUERY}(p)$, the two sets $\mu \text{QUERY}(p)$ and μNOGOOD . The problem solver receiving these sets will understand that $(A, B) \in \text{QUERY}(p)$ precisely when

1. there exists $(A', B') \in \mu \text{QUERY}(p)$:
 $(A', B') \preceq (A, B)$.
2. for all $(A', B') \in \text{NOGOOD}$: $(A', B') \not\preceq (A, B)$.
3. $A \subseteq A_{\max}, B \subseteq N, A \cap B = \emptyset$.

This third condition, implicit in the definition of ALL , is necessary to avoid the need for maximal elements of these sets.

Example 3: As in Example 1, let Σ comprise

$$\begin{aligned} a &\leftarrow \neg b \\ b &\leftarrow \neg c \end{aligned}$$

and let $A_{\max} = \emptyset$. Then

$$\begin{aligned} \mu \text{LABEL}(a) &= \{ (\emptyset, \{b\}) \} \\ \mu \text{LABEL}(b) &= \{ (\emptyset, \{c\}) \} \\ \mu \text{LABEL}(c) &= \emptyset. \end{aligned}$$

We have simply that $\mu \text{OUTASM}(p) = \{ (\emptyset, \{p\}) \}$ for $p = b, c$. Thus,

$$\begin{aligned} \mu \text{GOOD} &= \{ (\emptyset, \{c\}) \} \\ \mu \text{NOGOOD} &= \{ (\emptyset, \{b, c\}) \} \end{aligned}$$

and

$$\begin{aligned} \mu \text{QUERY}(a) &= \emptyset \\ \mu \text{QUERY}(b) &= \{ (\emptyset, \{c\}) \} \\ \mu \text{QUERY}(c) &= \emptyset. \end{aligned}$$

Note that $\text{QUERY}(a)$ is empty even though $\text{LABEL}(a)$ is not. This is because $\text{LABEL}(a) \cap \text{GOOD} = \text{NOGOOD}$ for this set of clauses.

Example 4: Now let Σ be from Example 2:

$$\begin{aligned} a &\leftarrow \neg b \\ b &\leftarrow \neg c \\ c &\leftarrow a. \end{aligned}$$

This time we set $A_{\max} = P = \{a, b, c\}$. We obtain

$$\begin{aligned} \mu \text{LABEL}(a) &= \{ (\emptyset, \{b\}), (\{a\}, \emptyset) \} \\ \mu \text{LABEL}(b) &= \{ (\emptyset, \{c\}), (\{b\}, \emptyset) \} \\ \mu \text{LABEL}(c) &= \{ (\emptyset, \{b\}), (\{a\}, \emptyset), (\{c\}, \emptyset) \} \end{aligned}$$

leading to

$$\begin{aligned} \mu \text{GOOD} &= \{ (\emptyset, \{b\}), (\emptyset, \{c\}), \\ &\quad (\{a, b\}, \emptyset), (\{b, c\}, \emptyset) \} \\ \mu \text{NOGOOD} &= \{ (\emptyset, \{b, c\}), (\{a\}, \{c\}) \}. \end{aligned}$$

The final queries are

$$\begin{aligned} \mu \text{QUERY}(a) &= \{ (\emptyset, \{b\}), (\{a, b\}, \emptyset) \} \\ \mu \text{QUERY}(b) &= \{ (\emptyset, \{c\}), (\{a, b\}, \emptyset), (\{b, c\}, \emptyset) \} \\ \mu \text{QUERY}(c) &= \{ (\emptyset, \{b\}), (\{a, b\}, \emptyset), (\{b, c\}, \emptyset) \}. \end{aligned}$$

4 IMPLEMENTATION

Examining the function `update-label` and the formulas for `BASE` and $\text{QUERY}(p)$, we can see that the entire algorithm of the NATMS involves standard set operations applied to various subsets of the set ALL . Therefore, an implementation of the NATMS is specified by the data structure used for representing subsets of ALL —i.e., sets of assumption set, stable base pairs—and the associated algorithms for set union, intersection, set difference and subset test. The NATMS must construct from scratch only the representation of $\text{INASM}(p)$ (the initial label of a proposition) and $\text{OUTASM}(p)$, defined in (15) and (16), respectively. All other quantities can be derived from set operations. OUT in `update-label` is generated as

$$\text{OUT} = \bigcap_{p \in \text{Neg}(\beta)} \text{OUTASM}(p).$$

In this section, we will outline two possible implementations of the NATMS: one patterned after de Kleer's implementation of the ATMS and the other patterned after the implementation described by [Pimentel & Cuadrado 89] for their stable model TMS.

4.1 MINIMAL SET IMPLEMENTATION

The ATMS operates directly on minimal proposition labels $\mu \text{LABEL}(p)$, where minimality in this case is defined with respect to ordinary set inclusion applied to assumption sets. Here we outline an analogous implementation of the NATMS using minimality with respect to \preceq .

In this approach, the representation of a set $X \subseteq \text{ALL}$ is simply a list of the elements in μX , each of which is a pair (A, B) . Generalizing from the ATMS, an efficient data structure for representing (A, B) is a pair of bit vectors defined over A_{\max} and N , respectively. A proposition $p \in A_{\max} \cap N$ will correspond to a bit in each vector, and we must constrain the data structure such that at least one of these bits is zero in order to guarantee $A \cap B = \emptyset$.

In terms of minimal sets, the representations of the two sets which must be constructed by the NATMS are simply

$$\begin{aligned} \mu \text{INASM}(p) &= \{ (\{p\}, \emptyset) \} \\ \mu \text{OUTASM}(p) &= \{ (\emptyset, \{p\}) \}. \end{aligned}$$

The following formulas can then be used in the label propagation algorithm and in the computation of GOOD and NOGOOD via (19), (20) and (22):

$$\begin{aligned} \mu(X \cup X') &= \mu(\mu X' \cup \mu X) \\ \mu(X \cap X') &= \mu\{ (A \cup A', B \cup B') \mid \\ &\quad (A, B) \in \mu X, (A', B') \in \mu X', \\ &\quad (A \cup A') \cap (B \cup B') = \emptyset \} \end{aligned}$$

$$\mu(X - X') = \{x \in \mu X \mid \text{for all } x' \in \mu X': x' \not\leq x\}$$

where $X, X' \subseteq \text{ALL}$ are upwardly closed with respect to \preceq . The test $(A, B) \preceq (A', B')$ and operations $A \cup A'$ and $B \cup B'$ required by these formulas can be performed efficiently as bit-by-bit computations on the bit-vector representations of A, A', B and B' .

4.2 DECISION TREE IMPLEMENTATION

While the minimal set representation is sufficient to represent upwardly closed sets, like $\text{LABEL}(p)$, it cannot represent ones which are not. We have developed a second representation, based on decision trees, which can represent arbitrary subsets of ALL , whether upwardly closed or not. Thus, the decision tree representation allows the direct computation of BASE and $\text{QUERY}(p)$ using (12) and (17).

Recall that a pair $(A, B) \in \text{ALL}$ has $A \subseteq A_{\max}, B \subseteq N, A \cap B = \emptyset$. Each (A, B) thus defines a partition of $A_{\max} \cup N$ into three disjoint sets: A, B and $A_{\max} \cup N - A - B$. A decision tree represents a set of pairs $X \subseteq \text{ALL}$ as a hierarchy of tests of various $p \in A_{\max} \cup N$ for membership in each partition. When applied to a given pair (A, B) , the final outcome of the tests will decide whether or not $(A, B) \in X$. Given a prescribed total ordering of $A_{\max} \cup N$, the tests can be ordered into a tree having at most three branches per node. If we require that the number of tests in the tree be minimum within the constraint of this ordering, the decision tree representing each subset of ALL will be unique.

Thus, a decision tree representing a set $X \subseteq \text{ALL}$ defines a boolean function $\text{decide}: \text{ALL} \rightarrow \{\text{false}, \text{true}\}$ which serves as the membership or *characteristic* function of the set. $\text{decide}(A, B)$ returns true if $(A, B) \in X$ and false if $(A, B) \notin X$. To achieve an efficient representation of decide , we constrain it to have one of six possible forms: decide-AB , decide-A , decide-B , decide-C , decide-T and decide-F . The last two of these perform the tests at the leaf of a decision tree and return either true or false unconditionally:

$\text{decide-T}(A, B)$
 true;

$\text{decide-F}(A, B)$
 false;

The remaining function types test a particular proposition $p \in A_{\max} \cup N$ for membership in one or more of the three partitions A, B or $C \equiv A_{\max} \cup N - A - B$, and calls another decision function depending on the result. The schemata for these functions are

$\text{decide-AB}(A, B)$
 if $p \in A$ then

```

decide-z(A, B);
else if  $p \in B$  then
  decide-y(A, B);
else
  decide-z(A, B);

decide-A(A, B)
if  $p \in A$  then
  decide-x(A, B);
else
  decide-y(A, B);

decide-B(A, B)
if  $p \in B$  then
  decide-x(A, B);
else
  decide-y(A, B);

decide-C(A, B)
if  $p \notin A$  and  $p \notin B$  then
  decide-x(A, B);
else
  decide-y(A, B);
```

In each function, p denotes one of the propositions from $A_{\max} \cup N$, while x, y and z each denote one of T, F, AB, A, B or C . The values given to p, x, y and z in a function will depend on the subset of ALL being represented and where in the decision tree the function is used. To achieve a unique, finite decision tree, we must impose the following restrictions on the recursive function references decide-x , etc. First, each referenced function must test a proposition higher than p in the ordering over $A_{\max} \cup N$. Second, the two or three functions which are referenced in a given function must all be different, i.e. be of different types or test different propositions. We also restrict which p can be tested in different function types as follows: $p \in A_{\max} \cap N$ for decide-AB , $p \in A_{\max}$ for decide-A and $p \in N$ for decide-B . Given these restrictions, the largest possible decision function tree would have $|A_{\max} \cup N|$ levels, with a branching factor of two or three at a level depending on whether the proposition tested at that level is in $A_{\max} \cap N$. Of course, the typical trees encountered in the computations by the NATMS will be much smaller than the worst case, particularly those representing sets which are upwardly closed.

The three primitive sets which the NATMS must construct are represented as follows. The empty set is represented by the decision function decide-F . $\text{INASM}(p)$ is represented by decide-F when $p \notin A_{\max}$ and by

$\text{decide-A}(A, B)$
 if $p \in A$ then
 decide-T(A, B);
 else
 decide-F(A, B);

when $p \in A_{\max}$. Similarly, OUTASM(p) is represented by decide-F when $p \notin N$ and otherwise by

decide-B(A, B)

```
if  $p \in B$  then
  decide-T( $A, B$ );
else
  decide-F( $A, B$ );
```

The algorithms which accomplish the set operations of union, intersection and set difference are too lengthy to show here. Each algorithm takes as input the decision function trees representing two arbitrary subsets of ALL, and returns the decision tree which represents the appropriate operation on the input sets. Each algorithm can be expressed recursively.

5 RELATED WORK

In this section, we review the work of other authors who have proposed truth maintenance systems related to ours. We begin by recalling Definition 3.1, in which an NATMS is defined as a quadruple $(P, A_{\max}, p_{\perp}, \Sigma)$. We will attempt to relate these other systems to this definition.

All of the systems we will consider allow Σ to contain nonmonotonic justifications, but only *some* allow assumption-based reasoning, i.e., $A_{\max} \neq \emptyset$. Section 5.1 and 5.2 consider systems which are *not* assumption-based while Section 5.3 considers one which, like ours, is assumption-based. As we will see, the systems discussed in Section 5.2 are special in that they make use of the ATMS through, roughly speaking, a mapping of nonmonotonicity into assumptions. We will state more precisely below why we do not classify these systems as assumption-based.

5.1 NONMONOTONIC SYSTEMS

The original TMS of [Doyle 79] is characterized by its ability to handle nonmonotonicity and contradictions, but not assumptions in the sense of [de Kleer 86]. In our notation, it is restricted to operate with $A_{\max} = \emptyset$ and therefore constitutes a special case of the NATMS with the form $(P, \emptyset, p_{\perp}, \Sigma)$. Doyle's system has also been classified as a "justification-based" system owing to the fact that, in logic programming terminology, it maintains a single stable model through a labeling of propositions with supporting justifications.

The algorithm of [Pimentel & Cuadrado 89], based on stable models, was intended as a complete implementation of the TMS of [Doyle 79], in the sense that it finds all stable models and always correctly terminates, even if no stable model exists. They compute the set of stable models using an equation essentially equivalent to (17), although the intersection was performed over P rather than N . This choice was made because The-

orem 2.1 was not known by the authors at that time. Because the method used in [Pimentel & Cuadrado 89] is subsumed by that of the present paper, it will not be discussed further.

A second nonmonotonic algorithm, also complete, is contained in [Junker & Konolige 90]. Junker and Konolige define their own notion of an "extension" of Σ . Their algorithm computes all extensions of Σ according to their definition. We show below that their extensions are in fact exactly the stable models, and that the functionality of their TMS is therefore identical to that of [Pimentel & Cuadrado 89] and subsumed by that of the present paper.

Before describing their TMS, we note that the bulk of Junker and Konolige's paper concerns a method for translating autoepistemic and default theories into TMS justifications.³ This allows *any* complete nonmonotonic TMS, including that of the present paper, to compute the extensions of an autoepistemic or default theory.

Junker and Konolige define extensions in terms of the transitive and reflexive closure of a certain operator. Using our notation, the operator is

$$\text{Apply}_{\Sigma, J}(I) = \{ \text{head}(\alpha) \mid \alpha \in \Sigma, \text{Pos}(\alpha) \subseteq I, \text{Neg}(\alpha) \cap J = \emptyset \}.$$

Junker and Konolige note that if Σ is monotonic, then $\text{Neg}(\alpha)$ is always empty, and so we may simply write $\text{Apply}_{\Sigma}(I)$ for $\text{Apply}_{\Sigma, \emptyset}(I)$ which is equal to $\text{Apply}_{\Sigma, J}(I)$ for any J .

By $\text{Apply}_{\Sigma, J}^*(I)$ we mean the minimal set that contains I and is closed with respect to $\text{Apply}_{\Sigma, J}$, i.e.,

$$\text{Apply}_{\Sigma, J}(\text{Apply}_{\Sigma, J}^*(I)) \subseteq \text{Apply}_{\Sigma, J}^*(I).$$

Junker and Konolige define I to be an *extension* of Σ if

$$I = \text{Apply}_{\Sigma, I}^*(\emptyset).$$

We want to show that the extensions of Σ , according to this definition, are precisely the stable models of Σ . We first state the following lemma.

Lemma 5.1 *If Σ is monotonic, then $\text{Apply}_{\Sigma}^*(\emptyset) = \text{Min}(\Sigma)$.*

Proof: The operator Apply_{Σ} can be rewritten as

$$\begin{aligned} \text{Apply}_{\Sigma}(I) &= \text{Apply}_{\Sigma, \emptyset}(I) \\ &= \{ \text{head}(\alpha) \mid \alpha \in \Sigma, \text{Pos}(\alpha) \subseteq I, \text{Neg}(\alpha) \cap \emptyset = \emptyset \} \\ &= \{ \text{head}(\alpha) \mid \alpha \in \Sigma, \text{Pos}(\alpha) \subseteq I \}. \end{aligned}$$

³The only limitations are that the theory must consist of a finite set of defaults and premises, and that derivability must be decidable for the base language [Junker & Konolige 90].

The reader will recognize the last expression as the familiar operator over the lattice of interpretations used in the standard fixed point semantics for logic programs [Lloyd 87]. $\text{Apply}_{\Sigma}^*(\emptyset)$ is precisely the least fixed point of this operator. The lemma follows from the equivalence between fixed point and minimal model semantics for monotonic logic programs [Lloyd 87]. ■

Theorem 5.1 I is an extension of Σ iff it is a stable model of Σ .

Proof: From the definition of the operator, we can check that $\text{Apply}_{\Sigma,I} = \text{Apply}_{\Sigma}$, for any I . Therefore, for any I , we have $\text{Apply}_{\Sigma,I}^*(\emptyset) = \text{Apply}_{\Sigma}^*(\emptyset) = \text{Min}(\Sigma_I)$, where the last equality follows from the lemma. Thus, $I = \text{Apply}_{\Sigma,I}^*(\emptyset)$ iff $I = \text{Min}(\Sigma_I)$ which establishes the theorem. ■

Since extensions as defined above are simply an alternative characterization of stable models, it follows that Junker and Konolige's algorithm is equivalent in its output to an NATMS of the restricted form $(P, \emptyset, p_{\perp}, \Sigma)$.

5.2 NONMONOTONIC SYSTEMS EMPLOYING THE ATMS

As stated in Section 3, de Kleer's ATMS may be described as an NATMS $(P, A_{\max}, p_{\perp}, \Sigma)$ with Σ restricted to Horn clauses. In other words, the ATMS requires $N = \emptyset$. Naturally, there has been considerable interest in relaxing this restriction so that some form of nonmonotonicity can be handled. However, recent efforts in this direction have not been entirely satisfactory.

Junker has presented a system [Junker 89] which maps a nonmonotonic TMS $(P, \emptyset, p_{\perp}, \Sigma)$ into de Kleer's ATMS and uses the ATMS with some postprocessing to find the stable models of Σ . (The NATMS, in contrast, finds the stable models of $\Sigma \cup A$ for various assumption sets A .) Although the ATMS is used internally, assumptions play no role in the input or output of Junker's system. His system is essentially a complete nonmonotonic TMS which happens to use the ATMS as a back-end processor. A simple way to see this is to observe that, in the case of purely monotonic justifications, Junker's system does not default to the behavior of a monotonic ATMS.

Let us now describe the precise correspondence between the system of [Junker 89] and the NATMS. Junker starts with a system $\Gamma = (P, \emptyset, p_{\perp}, \Sigma)$. For each $B \subseteq N$, define a set of new propositions

$$\text{Out}(B) = \{ \text{out}(p) \mid p \in B \}.$$

Given Σ defined over P , Junker constructs an augmented set of propositions P' , given by

$$P' = P \cup \text{Out}(N)$$

and a monotonic set of clauses Σ^* defined over P' , as follows

$$\begin{aligned} \Sigma' &= \{ c \leftarrow a_1, \dots, a_n, \text{out}(b_1), \dots, \text{out}(b_m) \mid \\ &\quad c \leftarrow a_1, \dots, a_n, \neg b_1, \dots, \neg b_m \in \Sigma \} \\ \Sigma^* &= \Sigma' \cup \{ p_{\perp} \leftarrow p, \text{out}(p) \mid p \in N \}. \end{aligned}$$

Using Σ^* , Junker's technique is to construct a new system Γ^* , given by $\Gamma^* = (P', \text{Out}(N), p_{\perp}, \Sigma^*)$. Since Σ^* contains Horn clauses only, Γ^* has the form of a monotonic ATMS. Thus, the ATMS receives an expanded set of propositions and clauses, compared to the original system, and uses the additional propositions, $\text{Out}(N)$, as assumptions. The assumptions handled by the ATMS thus actually correspond to the out-assumptions of the NATMS.

For any interpretation $I \subseteq P$, Junker distinguishes a subset of P' , $\text{MapOut}(I) = \text{Out}(I \cap N)$.⁴ Note that for any $B \subseteq N$, we have

$$\text{Out}(B) = \text{MapOut}(\overline{B}). \quad (23)$$

$\text{MapOut}(\overline{B})$ for $B \subseteq N$ are the sets which Γ^* treats as assumption sets, after the translation above. We observe that Theorem 1 of [Junker 89] can be stated as

$$\text{for all } p \in P: \Sigma_I \vdash p \text{ iff } \Sigma^* \cup \text{MapOut}(I) \vdash p. \quad (24)$$

Junker's algorithm is based on the computation of certain assumption sets in Γ^* he calls *extension bases*, which have a one-to-one correspondence with the stable models⁵ of Σ . We give a definition of extension bases equivalent to Junker's in our notation. Let $B \subseteq N$. Then $\text{Out}(B)$ is an extension base iff

$$\begin{aligned} \text{for all } \text{out}(p) \in \text{Out}(N): \\ \text{out}(p) \in \text{Out}(B) \text{ xor } \Sigma^* \cup \text{Out}(B) \vdash p. \end{aligned}$$

We can now state the following theorem.

Theorem 5.2 For all $B \subseteq N$, $\text{Out}(B)$ is an extension base iff B is a stable base.

Proof: The condition for $\text{Out}(B)$ to be an extension base can be restated as

$$\text{for all } p \in N: p \in B \text{ xor } \Sigma^* \cup \text{Out}(B) \vdash p,$$

which is equivalent to

$$\text{for all } p \in N: p \in \overline{B} \text{ iff } \Sigma^* \cup \text{Out}(B) \vdash p.$$

Using (23) and (24), this condition becomes

$$\text{for all } p \in N: p \in \overline{B} \text{ iff } \Sigma_{\overline{B}} \vdash p$$

⁴ Junker denotes this set $A(I)$. We have changed the notation to avoid confusion with sets A .

⁵ [Junker 89] uses default logic for his formalization of the TMS and therefore shows the correspondence with default extensions. However, the equivalent result holds for stable models as well, as shown in [Eshghi 90].

which is identical to Definition 2.1. ■

As a result, Theorem 2 of [Junker 89] is equivalent to our Theorem 2.1. The one-to-one correspondence between extension bases and stable models is equivalent to the one-to-one correspondence between stable bases and stable models. This implies that the translation from Σ to Σ^* and the introduction of $Out(N)$ as additional propositions are unnecessary. Given the label representation of the NATMS, all of the necessary computation can be done with the original nonmonotonic Σ defined over P .

Junker shows that all extension bases are supersets of the minimal assumption sets returned by Γ^* . However, the converse is not true: an assumption set contained in an ATMS label is not guaranteed to be a subset of an extension base, i.e., *valid* in Junker's terminology. Hence, Junker provides a non-incremental post-processing algorithm which takes a single assumption set and tests whether it is valid. The algorithm introduces choice points wherever multiple extensions can arise and uses backtracking to insure that they are all found.

Evidently the postprocessing performed by Junker's algorithm corresponds to the intersection of $LABEL(p)$ with $GOOD$ in (21), while our set $NOGOOD$ contains exactly the nogoods which Γ^* returns. We note this as a conjecture as we have not proved this formally.

We should note that [Eshghi 90] presents a variant of [Junker 89] using stable model semantics rather than default logic. Eshghi's system uses the same translation from Γ to Γ^* and employs a monotonic ATMS in the same way as Junker. One difference from Junker is that rather than test the individual members of labels, Eshghi employs an algorithm to directly compute the set of extension bases, which he calls *stable generators*. Again, we infer that this is closely related to the intersection of $LABEL(p)$ with $BASE$ in our system; however, the set-theoretic expression for $BASE$ in (17) is much simpler than Eshghi's nondeterministic algorithm.

In summary, although they use the ATMS internally, the systems of [Junker 89] and [Eshghi 90] are restricted to the form $(P, \emptyset, p_\perp, \Sigma)$. However, it appears likely that this restriction is not inherent to their approach and that their translation may work with $A_{\max} \cup Out(N)$ given to the ATMS. In effect, this is how the system of [Dressler 88] works, as we will see in the next section.

5.3 NONMONOTONIC, ASSUMPTION-BASED SYSTEMS

[Dressler 88] describes an extension to the ATMS which allows for nonmonotonicity in a manner similar to the NATMS. His system has been given semantics in [Reinfrank *et al.* 89] via a translation of justifications

into autoepistemic logic. We will adopt the treatment of [Reinfrank *et al.* 89] for the sake of comparison with the NATMS. By stating their definitions in terms of the stable model semantics, we can show that the semantics they propose for assumption-based systems is in fact equivalent to that of the NATMS.

[Reinfrank *et al.* 89] employ the same mapping of negated literals to $Out(N)$ as do [Junker 89] and [Eshghi 90]. As a result, it will be convenient to use some of the definitions of the previous section. The basic construct employed by [Reinfrank *et al.* 89] is the μ -extension. Given a quadruple $(P, A_{\max}, p_\perp, \Sigma)$ and an assumption set $A \subseteq A_{\max}$, $J \subseteq P \cup Out(N)$ is a μ -extension of $\Sigma' \cup A$ iff

$$J = \text{Min}(\Sigma' \cup A \cup \text{MapOut}(J)).$$

We observe that Lemma 4.5 of [Reinfrank *et al.* 89] can be stated as follows: given a quadruple $(P, A_{\max}, p_\perp, \Sigma)$ and an assumption set $A \subseteq A_{\max}$,

1. if I is a stable model of $\Sigma \cup A$, then $I \cup \text{MapOut}(I)$ is a μ -extension of $\Sigma' \cup A$, and
2. if J is a μ -extension of $\Sigma' \cup A$, then $J \cap P$ is a stable model of $\Sigma \cup A$.

We can now state the following theorem.

Theorem 5.3 J is a μ -extension of $\Sigma' \cup A$ iff $J = I \cup Out(B)$ for some stable model I and stable base B of $\Sigma \cup A$, where $B = \bar{I} \cap N$.

Proof: (\Rightarrow) Suppose J is a μ -extension of $\Sigma' \cup A$. By Reinfrank's Lemma 4.5, we know $I = J \cap P$ is a stable model of $\Sigma \cup A$. We may take

$$\begin{aligned} B &= \bar{I} \cap N \\ &= (\overline{J \cap P}) \cap N \\ &= \bar{J} \cap N. \end{aligned}$$

Since J is a μ -extension of $\Sigma' \cup A$, we know that $J = \text{Min}(\Sigma' \cup A \cup \text{MapOut}(J))$, which implies that

$$\begin{aligned} \text{MapOut}(J) &= Out(\bar{J} \cap N) \\ &= J \cap Out(N). \end{aligned}$$

Because $P \cap Out(N) = \emptyset$, we may write

$$\begin{aligned} J &= (J \cap P) \cup (J \cap Out(N)) \\ &= (J \cap P) \cup Out(\bar{J} \cap N) \\ &= I \cup Out(B). \end{aligned}$$

The last equality establishes the desired condition.

(\Leftarrow) Suppose I is a stable model and B a stable base of $\Sigma \cup A$, with $B = \bar{I} \cap N$. By Reinfrank's Lemma 4.5, we know that $J = I \cup \text{MapOut}(I)$ is a μ -extension of $\Sigma' \cup A$. But $\text{MapOut}(I) = Out(\bar{I} \cap N) = Out(B)$, so $J = I \cup Out(B)$ is a μ -extension of $\Sigma' \cup A$, as required. ■

6 CONCLUSIONS

We have presented a nonmonotonic ATMS which fully integrates abductive and nonmonotonic reasoning and is strictly more general than several previous efforts to integrate nonmonotonicity into the ATMS. Our system is based on a label propagation scheme analogous to that of the ATMS, where we have extended the label of a proposition and the result of a TMS query to include both assumption sets and stable bases, serving as the in-assumptions and out-assumptions which support a proposition.

By using stable bases, our TMS algorithm works entirely within the original set of justifications and propositions given to the system. A translation into an equivalent monotonic TMS is not required. Since stable bases are computed using only the propositions negated in a set of justifications, the computational impact of nonmonotonicity is isolated and minimized. We are looking for further ways to optimize the algorithm in special cases, such as when the set of justifications is *stratified* and must have a unique stable model.

The NATMS has been presented in set-theoretic terms, resulting in rather simple formulas for the computations performed by the system. We have shown how the implementation of the NATMS may be accomplished in a manner similar to the ATMS, using representations of labels and queries in terms of minimal sets and nogoods. We have also described an alternative implementation based on a more general representation of sets in terms of decision trees.

The semantics of our system is given precisely by the stable model semantics of logic programs. Recently, a number of authors have presented semantics for nonmonotonic TMS's in terms of autoepistemic logic [Fujiwara & Honiden 89; Reinfrank *et al.* 89, Elkan 90] and default logic [Junker 89]. Furthermore, [Junker & Konolige 90] have given the reverse translation (i.e., from autoepistemic and default logic into a nonmonotonic TMS). Therefore, the NATMS and other nonmonotonic TMS's offer a limited implementation of these other formalisms.

Acknowledgments

This work was partially supported by the Founding Members of the Earth Resources Laboratory, Massachusetts Institute of Technology.

References

- [de Kleer 86] J. de Kleer. An assumption-based TMS, *Artificial Intelligence* 28, 127-162, 1986.
- [Doyle 79] J. Doyle. A truth maintenance system, *Artificial Intelligence* 12, 231-272, 1979.

[Doyle 83] J. Doyle. Some theories of reasoned assumptions: an essay in rational psychology, *Technical Report CMU-CS-83-125*, Department of Computer Science, Carnegie-Mellon University, 1983.

[Dressler 88] O. Dressler. An extended basic ATMS, *Proceedings of the Second International Workshop on Nonmonotonic Reasoning*, LNCS 346, Springer-Verlag, 143-163, 1988.

[Elkan 90] C. Elkan. A rational reconstruction of nonmonotonic truth maintenance systems, *Artificial Intelligence* 43, 219-234, 1990.

[Eshghi 90] K. Eshghi. Computing stable models by using the ATMS, *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, MIT Press, 272-277, 1990.

[Fujiwara & Honiden 89] Y. Fujiwara & S. Honiden. Relating the TMS to autoepistemic logic, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Morgan Kaufmann, 1199-1205, 1989.

[Gelfond & Lifschitz 88] M. Gelfond & V. Lifschitz. The stable model semantics for logic programming, *Logic Programming: Proceedings of the Fifth International Conference*, MIT Press, 1070-1080, 1988.

[Junker 89] U. Junker. A correct non-monotonic ATMS, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Morgan Kaufmann, 1049-1054, 1989.

[Junker & Konolige 90] U. Junker & K. Konolige. Computing the extensions of autoepistemic and default logics with a truth maintenance system, *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, MIT Press, 278-283, 1990.

[Lloyd 87] J. Lloyd. *Foundations of Logic Programming*, Springer-Verlag, 1987.

[Moore 85] R. Moore. Semantical considerations on nonmonotonic logic, *Artificial Intelligence* 25, 75-94, 1985.

[Pimentel & Cuadrado 89] S. G. Pimentel & J. L. Cuadrado. A truth maintenance system based on stable models, *Logic Programming: Proceedings of the North American Conference*, MIT Press, 274-290, 1989.

[Reinfrank *et al.* 89] M. Reinfrank, O. Dressler & G. Brewka. On the relation between truth maintenance and autoepistemic logic, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Morgan Kaufmann, 1206-1212, 1989.

[Reiter & de Kleer 87] R. Reiter & J. de Kleer. Foundations of assumption-based truth maintenance systems: preliminary report, *Proceedings of the Sixth International Conference on Artificial Intelligence (AAAI-87)*, Morgan Kaufmann, 183-188, 1987.

Assertional Default Theories: A Semantical View

Torsten Schaub
 Fachgebiet Intellektik
 Technische Hochschule Darmstadt
 Alexanderstraße 10
 D-6100 Darmstadt, Germany
 e-mail: xiistsch@ddathd21.bitnet

Abstract

This paper provides a semantical characterization of Brewka's variant of default logic. The former can be seen as a realization of Poole's "commitment to assumptions" as well as Makinson's demand for cumulativity. Therefore, we present a model theoretic semantics that is subsequently evaluated and which is shown to provide useful insights into Brewka's formalism and Poole's and Makinson's notions.

1 INTRODUCTION

Reiter's *default logic* [1980] is one of the most natural approaches to default reasoning. Unfortunately, it lacks cumulativity¹, i.e. adding nonmonotonic theorems to the premises may alter the nonmonotonic theories [Makinson, 1989]. Written down formally, a theory operator Th is cumulative iff we have for arbitrary sets of formulas R and S that

$$R \subseteq S \subseteq \text{Th}(R) \implies \text{Th}(R) = \text{Th}(S).$$

Another problem has been addressed by Poole [1989], who shows that default logic does not stick to its assumptions in a proper way, i.e. a nonmonotonic theory may be justified by contradicting assumptions. Such inconsistencies may lead to unintuitive results.

Both of those "defects" have been attacked and solved by Brewka [1991] who introduced a slight variant of default logic, called *cumulative default logic*. But this logic lacks clear model theoretic semantics, since those proposed for default logic [Etherington, 1986; Etherington, 1987; de Guerreiro and Casanova, 1990] are not applicable any more.

In what follows we try to view briefly the formal definitions of default logic and its semantics. Afterwards we

¹See [Makinson, 1989] for a detailed discussion on cumulative inference relations.

introduce Brewka's variant and show how the former semantics has to be altered to cope with the variant. We argue that the presented semantics provides useful semantical insights into cumulative default logic as well as into nonmonotonic logics as such. We motivate this by presenting two examples that reflect the problems concerning commitment and cumulativity.

2 DEFAULT THEORIES AND THEIR SEMANTICS

A *closed default theory*, $\Delta = (D, W)$, consists of a set of closed first order formulas W and a set of *closed default rules* D . A default rule is any expression of the form

$$\frac{\alpha : \beta}{\omega},$$

where α, β and ω are closed first order formulas.² Thereby, α is called the *prerequisite*, ω the *consequent* and β the *justification* of the default rule. In what follows we will confine ourselves to closed default theories.

Using default rules, hypothetical knowledge is incorporated in the framework by using additional non-monotonic inference rules. They sanction inferences that rely upon given as well as absent knowledge and therefore could not be made in a classical framework. A default rule is applicable, if its prerequisite is known and its justification is *consistent*, i.e. a contradiction cannot be derived, if we would add its negation to our theory.

An *extension* of the initial set of facts is defined as all formulas derivable from the given knowledge using the classical inference rules and all specified default rules.

²For sake of simplicity we restrict ourselves to singular default rules, i.e. default rules with one justification. They have been advocated in [Reiter, 1987] and in cumulative default logic multiple justifications correspond to their conjunction.

Therefore, Reiter demands three properties to hold for an extension: it should contain our initial set of facts, it should be deductively closed and each consequent of each applicable default rule should be in it. How an extension can be characterized formally is described below.

Definition 2.1 Let $\Delta = (D, W)$ be a default theory. For any set S of formulas let $\Gamma_{DL}(S)$ be the smallest set such that

1. $W \subseteq \Gamma_{DL}(S)$,
2. $\text{Th}(\Gamma_{DL}(S)) = \Gamma_{DL}(S)$,
3. If $\frac{\alpha : \beta}{\omega} \in D$ and $\alpha \in \Gamma_{DL}(S)$ and $\neg\beta \notin S$ then $\omega \in \Gamma_{DL}(S)$.

A set of formulas E is an extension for Δ iff $\Gamma_{DL}(E) = E$.

Default logic deals with many commonsense examples very well, but it has the already mentioned problems: *non-commitment* and *non-cumulativity*. These are sketched by the following examples.

Example 2.1 (non-commitment)

The default theory

$$\Delta = \left(\left\{ \frac{: B}{C}, \frac{: \neg B}{D} \right\}, \emptyset \right)$$

has only one extension, $\text{Th}(\{C, D\})$. Both default rules have been fired, although they have contradicting justifications.

Example 2.2 (non-cumulativity³)

The default theory

$$\Delta = \left(\left\{ \frac{: A}{A}, \frac{A \vee B : \neg A}{\neg A} \right\}, \emptyset \right)$$

has one extension, $\text{Th}(\{A\})$. This extension inevitably contains $A \vee B$.

Adding the last disjunction to the premises yields the default theory

$$\Delta' = \left(\left\{ \frac{: A}{A}, \frac{A \vee B : \neg A}{\neg A} \right\}, \{A \vee B\} \right)$$

that has two extensions $\text{Th}(\{A\})$ and $\text{Th}(\{\neg A, B\})$.

No matter whether we are employing a sceptical or credulous notion of a theory. In both cases we obtain different ones.

³This example is originally due to David Makinson [Makinson, 1989].

Even though default logic has intuitively been well understood, it took several years until a model theoretic semantics was given. Etherington [1987; 1988] introduces a partial order \geq_δ on $2^{\text{MOD}(W)}$ that captures a default's preference for more specialized world descriptions.⁴ Unlike other approaches (e.g. circumscription [McCarthy, 1980]) that impose a *preference relation* on sets of models, here the same is done on the powerset.

Definition 2.2 Let $\delta = \frac{\alpha : \beta}{\omega}$ be a default rule, Π a set of models and $\Pi_1, \Pi_2 \in 2^\Pi$. The partial order \geq_δ is defined as follows.

$\Pi_1 \geq_\delta \Pi_2$ holds iff

1. $\forall \pi \in \Pi_2. \pi \models \alpha$,
2. $\exists \pi \in \Pi_2. \pi \models \beta$,
3. $\Pi_1 = \Pi_2 \setminus \{\pi \mid \pi \models \neg\omega\}$.

This order can be easily extended to the transitive order \geq_D that holds iff

$$\exists \delta \in D. \Pi_1 \geq_\delta \Pi_2.$$

Concerning normal default theories, it is sufficient to take into account the \geq_D -maximal elements of $2^{\text{MOD}(W)}$, whereas for others an additional so-called *stability condition* is necessary to prove soundness and completeness wrt the corresponding extensions.

Definition 2.3 Let $\Delta = (D, W)$ be a default theory and Π a \geq_D -maximal set of models out of $2^{\text{MOD}(W)}$. Π is called stable for Δ iff there is a set of default rules $D' \subseteq D$ such that

1. $\Pi \geq_{D'} \text{MOD}(W)$,
2. $\forall \delta = \frac{\alpha : \beta}{\omega} \in D'. \exists \pi \in \Pi. \pi \models \beta$.

Regarding the semantics, we now see why default theories do not stick to their assumptions: Together, the second condition of the former and the latter definition require only one model to satisfy the justification of a considered default rule. But it is not required, that there has to be one that satisfies all of the justifications of the default rules used during a derivation. Hence, concerning the justifications default logic preserves only a kind of *distributed consistency*.

This becomes obvious, if we take a closer look at the model structure obtained in Example 2.1. Looking at the induced ordering illustrated in Figure 1, we observe one stable set of models,

$$\text{MOD}(\{C \wedge D\}).$$

⁴ $\text{MOD}(W)$ denotes the set of all models of W .

But this set is properly divided into two sets of models: those satisfying B and those satisfying $\neg B$. Even though both default rules have been applied there is no model satisfying both of the corresponding justifications.

Regarding Example 2.2 and its semantical background as illustrated in Figure 2, we see that adding the non-monotonic theorem $A \vee B$ also enlarges the considered set of models, here $\text{MOD}(\{A\})$. So, which information has been lost?

In both examples, the unintuitive results occur because the implicit consistency assumptions have been lost. We are dealing with nonmonotonic theorems but we are not aware of the underlying assumptions.

3 ASSERTIONAL DEFAULT THEORIES

Brewka [1991] solved the two problems by strengthening the applicability condition for default rules and making the reasons for believing something explicit. In order to keep track of the assumptions syntactically, he introduced so-called *assertions*, i.e. formulas marked with the set of justifications and consequents of the default rules that have been used during their derivation.

Definition 3.1 Let $\alpha, \gamma_1, \dots, \gamma_m$ be formulas. An assertion ξ^5 is any expression of the form

$$\langle \alpha, \{\gamma_1, \dots, \gamma_m\} \rangle,$$

where $\alpha = \text{Form}(\xi)$ is called the asserted formula and the set $\{\gamma_1, \dots, \gamma_m\} = \text{Supp}(\xi)$ is called the support of α .⁶

To guarantee the proper propagation of the supports, the classical inference relation has to be extended. Let $\widehat{\text{Th}}$ denote the extended theory operator. If $\xi_1, \dots, \xi_n \in \widehat{\text{Th}}(\mathcal{S})$ and $\text{Form}(\xi_1), \dots, \text{Form}(\xi_n) \vdash \gamma$ then $\langle \gamma, \text{Supp}(\xi_1) \cup \dots \cup \text{Supp}(\xi_n) \rangle \in \widehat{\text{Th}}(\mathcal{S})$.

Now, an *assertional default theory* becomes a pair $\Lambda = (D, \mathcal{W})$, where D is a set of default rules and \mathcal{W} is a set of assertions. Furthermore, we assume them to be *well based*, i.e. that $\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W})$ are consistent. Accordingly an *assertional extension* is defined as follows.

Definition 3.2 Let $\Lambda = (D, \mathcal{W})$ be an assertional default theory. For any set \mathcal{S} of assertions let $\Gamma_{CDL}(\mathcal{S})$ be the smallest set fulfilling the following three properties.

⁵If not stated explicitly, we denote an assertion as $\xi = \langle \alpha, \{\gamma_1, \dots, \gamma_m\} \rangle$.

⁶The two projections can naturally be extended to sets of assertions.

1. $\mathcal{W} \subseteq \Gamma_{CDL}(\mathcal{S})$,
2. $\widehat{\text{Th}}(\Gamma_{CDL}(\mathcal{S})) = \Gamma_{CDL}(\mathcal{S})$,
3. If $\frac{\alpha : \beta}{\omega} \in D$ and $\langle \alpha, \text{Supp}(\alpha) \rangle \in \Gamma_{CDL}(\mathcal{S})$ and $\text{Form}(\mathcal{S}) \cup \text{Supp}(\mathcal{S}) \cup \{\beta\} \cup \{\omega\} \not\vdash \perp$ then $\langle \omega, \text{Supp}(\alpha) \cup \{\beta\} \cup \{\omega\} \rangle \in \Gamma_{CDL}(\mathcal{S})$.

A set of assertions \mathcal{E} is an assertional extension for Λ iff $\Gamma_{CDL}(\mathcal{E}) = \mathcal{E}$.

Comparing the last definition with that of Reiter, we observe two basic differences:

1. The applicability condition of a default rule requires the consistency of its justification and its consequent with all asserted formulas as well as all supports of an assertional extension.
2. The justifications and consequents of all generating default rules are recorded.

Additionally, assertional default theories share several nice properties: the *existence of extensions* is guaranteed, they are *semimonotonic*⁷ and all extensions of a given assertional default theory are *weakly orthogonal*⁸ to each other. Due to their ability to keep track of the assumptions made, they are able to cope with the problems already mentioned.

Example 3.1 (commitment)
The assertional default theory

$$\Delta = \left(\left\{ \frac{\vdash B}{C}, \frac{\vdash \neg B}{D} \right\}, \emptyset \right)$$

has now two extensions, $\widehat{\text{Th}}(\{(C, \{B, C\})\})$ and $\widehat{\text{Th}}(\{(D, \{\neg B, D\})\})$.

Example 3.2 (cumulativity)
The assertional default theory

$$\Lambda = \left(\left\{ \frac{\vdash A}{A}, \frac{A \vee B : \neg A}{\neg A} \right\}, \emptyset \right)$$

has still one extension, $\widehat{\text{Th}}(\{(A, \{A\})\})$. It contains now the assertion $(A \vee B, \{A\})$.

Consequently, adding this to the premises yields the assertional default theory

$$\Lambda = \left(\left\{ \frac{\vdash A}{A}, \frac{A \vee B : \neg A}{\neg A} \right\}, \{(A \vee B, \{A\})\} \right)$$

that has still the same extension and no other.

⁷I.e. monotonic wrt to the default rules.

⁸I.e. at least the supports of two extensions contradict each other.

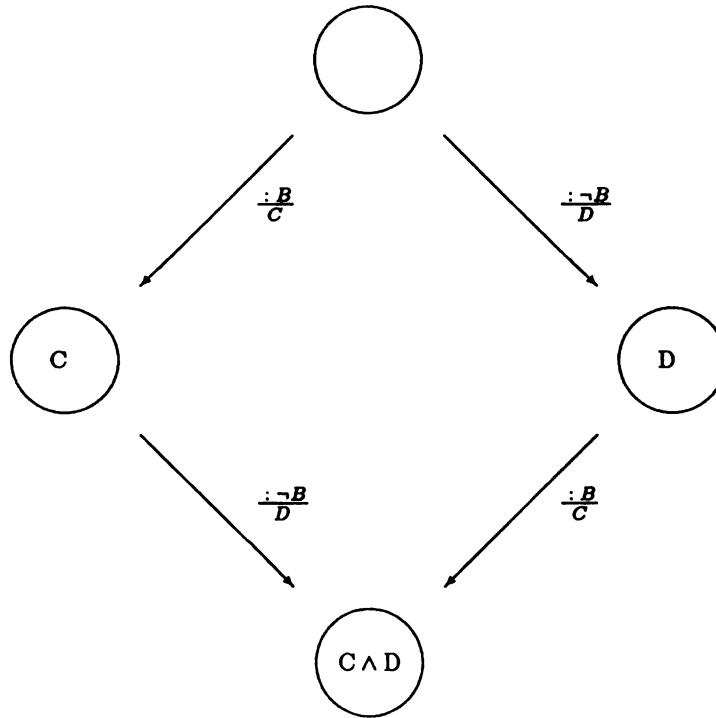
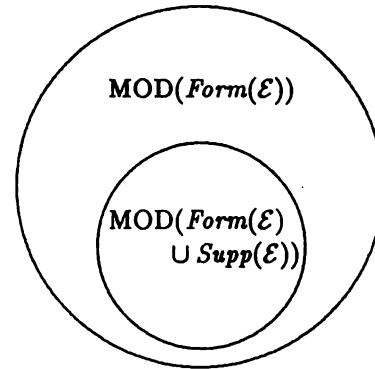


Figure 1: Non-commitment in default logic.

We see how easily we obtain the desired effects if we keep track of the justifications and the formulas derived using default rules. But how can this “syntactic sugar” be realized semantically?

4 A SEMANTICS FOR ASSERTIONAL DEFAULT THEORIES

In order to characterize assertional default theories and their behaviour a similar preference relation can now be defined by considering pairs like $(\Pi, \tilde{\Pi})$, where Π and $\tilde{\Pi}$ are sets of models. The intuition behind such a semantical structure is as follows. Viewing the justifications more as a kind of *working assumptions* the previously mentioned distributed consistency of default theories is not adequate any longer. Rather, we need to *focus* on those models that satisfy our assumptions. But since we cannot prove their validity there have to exist models that falsify them and “overlap” our focused models concerning our working assumptions. So, we just impose more structure on the sets of models under consideration, viewing the second component $\tilde{\Pi}$ — that is just a subset of Π — as our focused set of models. Given an assertional extension \mathcal{E} , we can illustrate the corresponding semantical structure

Figure 3: The model structure of an assertional extension \mathcal{E} .

as done in Figure 3.

Formally, we can achieve this by defining an appropriate order relating the consistency of the justifications with their validity in the focused models.

Definition 4.1 Let $\delta = \frac{\alpha : \beta}{\omega}$ be a default rule, Π a set of models and $\Pi_1, \Pi_2, \tilde{\Pi}_1, \tilde{\Pi}_2 \in 2^\Pi$. The partial order \succeq_δ is defined as follows.

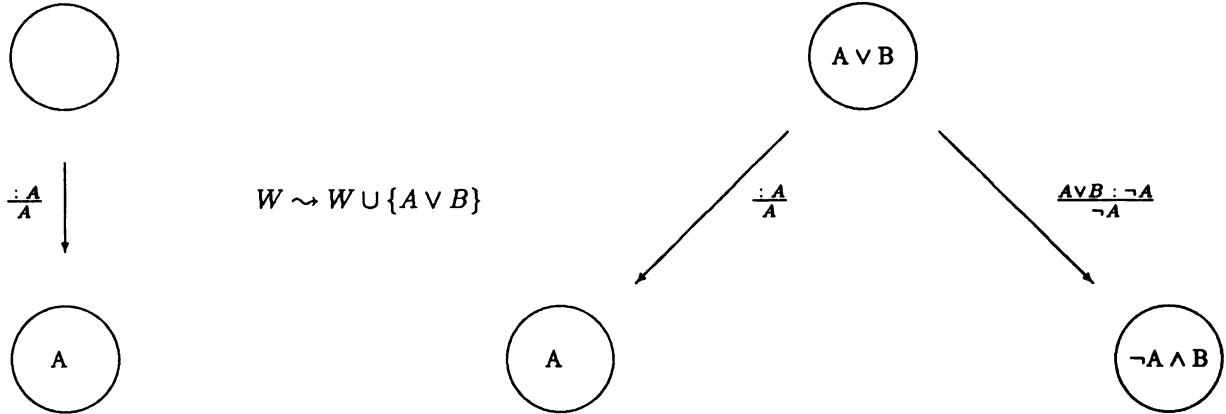


Figure 2: Non-cumulativity in default logic.

$(\Pi_1, \check{\Pi}_1) \succeq_{\delta} (\Pi_2, \check{\Pi}_2)$ holds iff

1. $\forall \pi \in \Pi_2. \pi \models \alpha$,
2. $\exists \pi \in \check{\Pi}_2. \pi \models \{\beta\} \cup \{\omega\}$,
3. $\Pi_1 = \Pi_2 \setminus \{\pi \mid \pi \models \neg \omega\}$,
4. $\check{\Pi}_1 = \check{\Pi}_2 \setminus \{\pi \mid \pi \models \neg \beta \text{ or } \pi \models \neg \omega\}$.

The induced ordering \succeq_D is defined analogously to that in Section 2. But using a strengthened consistency condition by requiring that all justifications and consequents have to be fulfilled by our focused models, we do not need a stability condition anymore. Technically, this is due to the fact that we are dealing with a semimonotonic default logic. But from the viewpoint of our focused models, we allow only those default rules to be applied that are compatible with them.

Afterall, we can provide the following correctness and completeness theorems.

Theorem 4.1 (Correctness)

Let $\Lambda = (D, \mathcal{W})$. If \mathcal{E} is an assertional extension of Λ then

$$(\text{MOD}(\text{Form}(\mathcal{E})), \text{MOD}(\text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E})))$$

is a \succeq_D -maximal element such that

$$\begin{aligned} &(\text{MOD}(\text{Form}(\mathcal{E})), \text{MOD}(\text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E}))) \succeq_D \\ &(\text{MOD}(\text{Form}(\mathcal{W})), \text{MOD}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W}))). \end{aligned}$$

Theorem 4.2 (Completeness)

Let $\Lambda = (D, \mathcal{W})$. If $(\Pi, \check{\Pi})$ is a \succeq_D -maximal element such that

$$\begin{aligned} &(\Pi, \check{\Pi}) \succeq_D \\ &(\text{MOD}(\text{Form}(\mathcal{W})), \text{MOD}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W}))) \end{aligned}$$

then there is an extension \mathcal{E} of Λ such that

$$\Pi \models \text{Form}(\mathcal{E}) \text{ and } \check{\Pi} \models \text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E}).$$

The proofs of the theorems are given in Appendix A. They are analogous to those given in [Etherington, 1988] and employ an inductive argumentation.

Again, concerning normal assertional default theories we obtain a simpler semantical characterization, provided that we are dealing with non supported base sets. In this case the larger set of models collapses to the focussed ones, since normal default rules require their justifications to be fulfilled after they have been proven to be consistent. Nevertheless, the need to maintain a semantical structure involving a focus arises, when making nonmonotonic theorems explicit. Otherwise, the underlying assumptions will be lost.

The way, our former examples are now accomplished semantically, is illustrated by the figures 4 and 5. Thereby, Figure 4 refers to Example 3.1 wheras Figure 5 illustrates Example 3.2.

Regarding Figure 4, we see why in Example 3.1 none of the other default rules is applicable any longer: The focus does not fulfill their justifications. Applying one of the default rules does not just add the consequents. It also makes us aware of the underlying assumptions. Adding C under the assumption that B is consistent prohibits us from assuming $\neg B$ to be consistent also, or vice versa.

Regarding Figure 5, we see why the second default rule in Example 3.2 is blocked, even though we have added the assertion $\langle A \vee B, \{A\} \rangle$ to the premises: Asserting

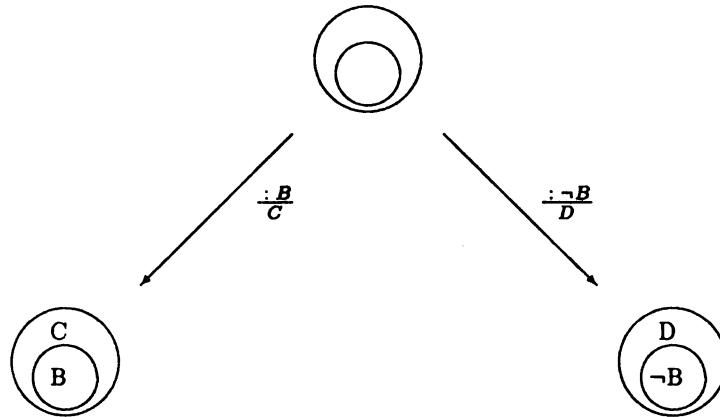


Figure 4: Commitment in cumulative default logic.

$A \vee B$ by focusing those models satisfying A does not allow $\neg A$ to be consistently assumed. Once again, we have been aware of the underlying assumptions.

In addition, the semantics also supplies us with insights into the nice properties of well based assertional default theories mentioned in Section 3. The existence of model structures is guaranteed, since Definition 4.1 ensures that $\tilde{\Pi}$ never becomes an empty set. The same definition also takes care of semimonotonicity. Because there has to exist a focussed model satisfying the prerequisite, consequent and the justification of an added default rule before it is going to be applied. Weak orthogonality is mirrored by the fact that there never exists a model which is shared by two different focussed sets of models.

5 DISCUSSION

Aside the semantical insights described so far, the semantical reflection provides us also the conjecture that the very strong syntactical relation between an asserted formula and its support is not really necessary. Hence, we conjecture that an equivalent formulation of cumulative default logic using first order formulas and an appropriate two-place fixed point operator à la [Łukasiewicz, 1988] — but with a different consistency condition — would work also. But we have not worked it out yet.

On the other hand we have seen that adding a non-monotonic theorem to the facts forces us also to add the underlying assumptions, if we want to preserve cumulativity. Consequently, the latter have to be filtered out before adding the nonmonotonic theorem. So, there is a need to have the specific supports.

Recently, a problem referred to as “floating conclusions” [Makinson and Brewka, 1991] arose when deal-

ing with sceptical reasoning. This is due to the fact that meeting two assertional extension does not produce anything about an asserted formula, if it occurs in both extension but with a different support.

Example 5.1 (floating conclusions)
The assertional default theory

$$\Lambda = \left(\left\{ \frac{: \neg B}{A}, \frac{: \neg A}{B} \right\}, \{A \rightarrow C, B \rightarrow C\} \right)$$

has two extensions, $\widehat{\text{Th}}(\{(A, \{\neg B, A\}), (C, \{\neg B, A\})\})$ and $\widehat{\text{Th}}(\{(B, \{\neg A, B\}), (C, \{\neg A, B\})\})$. Now, employing a sceptical notion of a theory we can draw no conclusion about C .

Looking at the semantics, we obtain two different model structures

$$(\text{MOD}(\{A, C\}), \text{MOD}(\{A, C, \neg B\}),)$$

and

$$(\text{MOD}(\{B, C\}), \text{MOD}(\{B, C, \neg A\}),).$$

Now, combining both model structures by unioning the focuses and the whole set of models separately yields a semantical structure

$$(\text{MOD}(\{C\}) \setminus \text{MOD}(\{\neg A \wedge \neg B\}), \\ \text{MOD}(\{C\}) \setminus \text{MOD}(\{A \leftrightarrow B\}))$$

all of whose models satisfy C .

The semantical structure corresponds to an assertional default theory including the assertions

$$\langle A \vee B, \{\neg B \wedge A \vee \neg A \wedge B\}, \\ \langle C, \{\neg B \wedge A \vee \neg A \wedge B\} \rangle.$$

This is the solution our semantics would yield. So, why

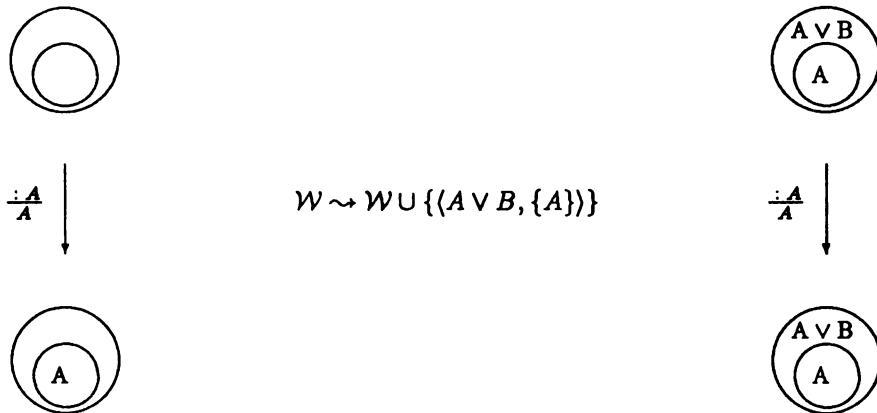


Figure 5: Cumulativity in cumulative default logic.

not adopting a corresponding mechanism of sceptical reasoning?

However, we have not yet investigated this kind of sceptical reasoning. Hence, it remains future work to look for the consequences (e.g. we have not yet proven that cumulativity is preserved).

Moreover, it remains an open question how Makinson's generalized variant of assertional default theories (described in [Brewka, 1991]) can be realized semantically. Instead of assertions he introduced so-called *affirmations*, i.e. formulas marked separately with the set of justifications and consequents of the default rules that have been used during their derivation. Using affirmations he was able to define a version of default logic that is cumulative but does not care about inconsistencies between justifications.

Last but not least, the question arises why we have adopted Etherington's semantics [1986; 1987] and not another one, e.g. [de Guerreiro and Casanova, 1990]? Both semantics yield the same set of models, but differ in the way the preference relation is defined. Hence, we have chosen the one that exposes our problems in a better way. Etherington's preference relation reflects exactly the transitions between theories or sets of models, respectively, induced by the corresponding default rules. We wanted to show how this process is envisaged in cumulative default logic. However, we think that a similar technique, i.e. requiring consistency wrt those models fulfilling all asserted formulas as well as their supports, also works for their semantics.

6 CONCLUSIONS

Human commonsense reasoning is strongly based on the ability to reason upon non verifiable assumptions

or better working assumptions. But people do not assume things arbitrarily, rather they keep track of their assumptions and at least verify that they do not contradict each other. No one would justify a conclusion by an assumption as well as its opposite.

Moreover, we argue that people who assume properties also assume their validity. In other words they focus on a certain class of models that satisfy their assumptions. Hence, concerning default logic we strengthen the meaning of justifications towards unverified reasons for believing something.

This becomes obvious, if we regard the default theory

$$\Delta = \left(\left\{ \frac{A : B}{C} \right\}, \{A\} \right).$$

In default logic's semantics we obtain a *flat* set of models

$$\Pi = \{\pi \mid \pi \models \{A, B, C\}\} \dot{\cup} \{\pi \mid \pi \models \{A, \neg B, C\}\},$$

whereas our semantics yields a pair

$$(\Pi, \check{\Pi}) = (\{\pi \mid \pi \models \{A, C\}\}, \{\pi \mid \pi \models \{A, \neg B, C\}\})$$

that corresponds to a *structured* set of models including a *focus*.

Hence, we admit more structured sets of models by focusing on those models that fulfill our assumptions. As we have seen this semantically accounts for "commitment to assumptions" and also leads to cumulativity.

A PROOFS OF THEOREMS

In the proofs we will use the terms *semantical* or *model structure*, respectively, to refer to the previously discussed pairs of models, $(\Pi, \check{\Pi})$.

Moreover, we need the following definition.

Definition A.1 *The set of generating default rules of an assertional extension \mathcal{E} of an assertional default theory Λ is defined as follows*

$$GD(\mathcal{E}, \Lambda) = \{\delta = \frac{\alpha : \beta}{\omega} \mid (\alpha, Supp(\alpha)) \in \mathcal{E}, Form(\mathcal{E}) \cup Supp(\mathcal{E}) \cup \{\beta\} \cup \{\omega\} \not\models \perp\}$$

Now, an assertional extension is characterized unambiguously by the set of generating default rules.

Lemma A.1 *Let Λ , \mathcal{E} and $GD(\mathcal{E}, \Lambda)$ be defined as usual. We have*

$$\mathcal{E} = \widehat{Th}(\mathcal{W} \cup \{(\omega, Supp(\alpha) \cup \{\beta\} \cup \{\omega\}) \mid (\alpha, Supp(\alpha)) \in \mathcal{E}, \frac{\alpha : \beta}{\omega} \in GD(\mathcal{E}, \Lambda)\}).$$

Proof A.1 See [Schaub, 1991].

A.1 PROOF OF CORRECTNESS

Definition A.2 *Let Λ , \mathcal{E} and $GD(\mathcal{E}, \Lambda)$ be defined as usual. Using the well ordered set of default rules*

$$GD(\mathcal{E}, \Lambda) = \{\delta_1, \delta_2, \delta_3, \delta_4, \dots\}$$

we can define a sequence of semantical structures as follows:

$$\begin{aligned} (\Pi_0, \check{\Pi}_0) &= \\ &(MOD(Form(\mathcal{W})), MOD(Form(\mathcal{W}) \cup Supp(\mathcal{W}))) \\ (\Pi_{i+1}, \check{\Pi}_{i+1}) &= \\ &(\{\pi \in \Pi_i \mid \pi \models \omega_i\}, \{\pi \in \check{\Pi}_i \mid \pi \models \beta_i, \omega_i\}). \end{aligned}$$

Where for each $i \geq 1$ we define ζ_i to be the default rule

$$\zeta_i = \delta_k = \frac{\alpha_k : \beta_k}{\omega_k} \in GD(\mathcal{E}, \Lambda),$$

with

$$\begin{aligned} k = \min\{j \mid \delta_j &= \frac{\alpha_j : \beta_j}{\omega_j}, \\ \delta_j &\notin \{\zeta_1, \dots, \zeta_i\}, \\ \Pi_i &\models \alpha_j, \\ \exists \pi \in \check{\Pi}_i. \pi &\models \beta_j, \omega_j\}. \end{aligned}$$

We observe that the sequence of the semantical structures $(\Pi_i, \check{\Pi}_i)$ is well defined and that each of the default rules $\zeta_i \in GD(\mathcal{E}, \Lambda)$ is chosen only once.

Lemma A.2 *Let Λ , \mathcal{E} and $GD(\mathcal{E}, \Lambda)$ be defined as usual. Let $(\Pi_0, \check{\Pi}_0), (\Pi_1, \check{\Pi}_1), (\Pi_2, \check{\Pi}_2), \dots$ be a sequence of semantical structures as defined in Definition A.2 and*

$$(\Pi, \check{\Pi}) = (\bigcap_{i=0}^{\infty} \Pi_i, \bigcap_{i=0}^{\infty} \check{\Pi}_i),$$

we have

1. $(\Pi_{i+1}, \check{\Pi}_{i+1}) \succeq_{\zeta_{i+1}} (\Pi_i, \check{\Pi}_i), \quad i \geq 0.$
2. $(\Pi, \check{\Pi}) \succeq_{GD(\mathcal{E}, \Lambda)} (MOD(Form(\mathcal{W})), MOD(Form(\mathcal{W}) \cup Supp(\mathcal{W}))),$
3. $\Pi \models Th(Form(\mathcal{W}) \cup Conseq(GD(\mathcal{E}, \Lambda))),$
4. $\check{\Pi} \models Th(Form(\mathcal{W}) \cup Justif(GD(\mathcal{E}, \Lambda)) \cup Conseq(GD(\mathcal{E}, \Lambda))).$

Proof A.2

1. By definition.

2. Proof by induction over i .

$$i = 0$$

$$\begin{aligned} (\Pi_0, \check{\Pi}_0) &\succeq_0 \\ &(MOD(Form(\mathcal{W})), MOD(Form(\mathcal{W}) \cup Supp(\mathcal{W}))) \end{aligned}$$

$$i \sim i + 1$$

$$\begin{aligned} (\Pi_{i+1}, \check{\Pi}_{i+1}) &\succeq_{\zeta_{i+1}} \\ &(\Pi_i, \check{\Pi}_i) \succeq_{\{\zeta_1, \dots, \zeta_i\}} \\ &(MOD(Form(\mathcal{W})), MOD(Form(\mathcal{W}) \cup Supp(\mathcal{W}))) \end{aligned}$$

Now, since \succeq_D is transitive, we obtain

$$\begin{aligned} (\Pi_{i+1}, \check{\Pi}_{i+1}) &\succeq_{\{\zeta_1, \dots, \zeta_i, \zeta_{i+1}\}} \\ &(MOD(Form(\mathcal{W})), MOD(Form(\mathcal{W}) \cup Supp(\mathcal{W}))). \end{aligned}$$

3. + 4. Proof by induction over i .

$$i = 0 \text{ Trivial.}$$

$$i \sim i + 1 \text{ Let}$$

$$\zeta_{i+1} = \frac{\alpha_{i+1} : \beta_{i+1}}{\omega_{i+1}}.$$

The induction hypotheses say that

$$\Pi_i \models Th(Form(\mathcal{W}) \cup \{\omega_{\zeta_1}, \dots, \omega_{\zeta_i}\}),$$

and

$$\begin{aligned} \check{\Pi}_i &\models Th(Form(\mathcal{W}) \cup \\ &\{\beta_{\zeta_1}, \dots, \beta_{\zeta_i}\} \cup \{\omega_{\zeta_1}, \dots, \omega_{\zeta_i}\}), \end{aligned}$$

where ω_{ζ_k} is the consequence of ζ_k .

By definition we have

$$\begin{aligned} (\Pi_{i+1}, \check{\Pi}_{i+1}) &= \\ &(\{\pi \in \Pi_i \mid \pi \models \omega_i\}, \{\pi \in \check{\Pi}_i \mid \pi \models \beta_i, \omega_i\}) \end{aligned}$$

that yields the claimed proposition.

Now, we can prove the following correctness theorem.

Theorem A.1 (Correctness)

Let $\Lambda = (D, \mathcal{W})$. If \mathcal{E} is an assertional extension of Λ then

$$(MOD(Form(\mathcal{E})), MOD(Form(\mathcal{E}) \cup Supp(\mathcal{E})))$$

is a \succeq_D -maximal element such that

$$(\text{MOD}(\text{Form}(\mathcal{E})), \text{MOD}(\text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E}))) \succeq_D (\text{MOD}(\text{Form}(\mathcal{W})), \text{MOD}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W}))).$$

According to Lemma A.1 we have

$$\begin{aligned} \mathcal{E} &= \widehat{\text{Th}}(\mathcal{W} \cup \{\langle \omega, \text{Supp}(\alpha) \cup \{\beta\} \cup \{\omega\} \rangle | \\ &\quad \langle \alpha, \text{Supp}(\alpha) \rangle \in \mathcal{E}, \frac{\alpha : \beta}{\omega} \in \text{GD}(\mathcal{E}, \Lambda)\}). \end{aligned}$$

Let

$$\begin{aligned} (\Pi_\epsilon, \check{\Pi}_\epsilon) &= \\ &(\text{MOD}(\text{Form}(\mathcal{E})), \text{MOD}(\text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E}))). \end{aligned}$$

We have to show the following two properties.

(a) There exists a set of default rules D' such that

$$\begin{aligned} (\Pi_\epsilon, \check{\Pi}_\epsilon) &\succeq_{D'} \\ &(\text{MOD}(\text{Form}(\mathcal{W})), \text{MOD}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W}))), \end{aligned}$$

(b) $(\Pi_\epsilon, \check{\Pi}_\epsilon)$ is a \succeq_D -maximal semantical structure.

We can distinguish the following two cases:

1. $\text{GD}(\mathcal{E}, \Lambda) = \emptyset$.

Let $D' = \text{GD}(\mathcal{E}, \Lambda) = \emptyset$. Hence, we have $\mathcal{E} = \widehat{\text{Th}}(\mathcal{W})$. Now, we have to regard the following two cases.

(a) $(\Pi_\epsilon, \check{\Pi}_\epsilon) \succeq_{\text{GD}(\mathcal{E}, \Lambda)} (\text{MOD}(\text{Form}(\mathcal{W})), \text{MOD}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W})))$, since $\text{GD}(\mathcal{E}, \Lambda) = \emptyset$.

(b) Now, assume $(\Pi_\epsilon, \check{\Pi}_\epsilon)$ is not \succeq_D -maximal. Then there exists a default rule

$$\delta' = \frac{\alpha' : \beta'}{\omega'} \in D,$$

so that

- i. $\Pi_\epsilon \models \alpha'$,
- ii. $\exists \pi \in \check{\Pi}_\epsilon. \pi \models \beta', \omega'$.

Since $\Pi_\epsilon \models \text{Form}(\mathcal{E})$ and $\Pi_\epsilon \supseteq \check{\Pi}_\epsilon \models \text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E})$ we have

- i. $\langle \alpha', \text{Supp}(\alpha') \rangle \in \mathcal{E}$,
- ii. $\text{Form}(\mathcal{E}) \cup \{\beta'\} \cup \{\omega'\} \not\models \perp$,

such that

$$\delta' \in \text{GD}(\mathcal{E}, \Lambda).$$

Contradiction to $\text{GD}(\mathcal{E}, \Lambda) = \emptyset$.

2. $\text{GD}(\mathcal{E}, \Lambda) \neq \emptyset$.

Let $D' = \text{GD}(\mathcal{E}, \Lambda)$. Again, we have to regard the following two cases.

(a) $(\Pi_\epsilon, \check{\Pi}_\epsilon) \succeq_{\text{GD}(\mathcal{E}, \Lambda)} (\text{MOD}(\text{Form}(\mathcal{W})), \text{MOD}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W})))$.

With Lemma A.2 we get the desired result.

(b) To prove that $(\Pi_\epsilon, \check{\Pi}_\epsilon)$ is \succeq_D -maximal we regard a default rule $\delta \in D$.

If $\delta \in \text{GD}(\mathcal{E}, \Lambda)$ we are done.

If $\delta \notin \text{GD}(\mathcal{E}, \Lambda)$ we get either

$$\langle \alpha, \text{Supp}(\alpha) \rangle \notin \mathcal{E}$$

or

$$\text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E}) \cup \{\beta\} \cup \{\omega\} \vdash \perp.$$

So, either $\Pi \not\models \alpha$ or $\exists \pi \in \check{\Pi}. \pi \models \beta, \omega$ and $(\Pi_\epsilon, \check{\Pi}_\epsilon)$ is \succeq_D -maximal.

A.2 PROOF OF COMPLETENESS

Definition A.3 Let $(\Pi, \check{\Pi})$ a model structure für Λ . Without loss of generality let

$$D' = \{\delta_1, \delta_2, \delta_3, \delta_4, \dots\}$$

be a minimal, well ordered set of default rules, so that

$$(\Pi, \check{\Pi}) \succeq_{D'}$$

$$(\text{MOD}(\text{Form}(\mathcal{W})), \text{MOD}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W}))).$$

We can define a sequence of semantical structures analogous to that in Definition A.2.

$$\begin{aligned} (\Pi_0, \check{\Pi}_0) &= \\ &(\text{MOD}(\text{Form}(\mathcal{W})), \text{MOD}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W}))) \end{aligned}$$

$$\begin{aligned} (\Pi_{i+1}, \check{\Pi}_{i+1}) &= \\ &(\{\pi \in \Pi_i \mid \pi \models \omega_i\}, \{\pi \in \check{\Pi}_i \mid \pi \models \beta_i, \omega_i\}). \end{aligned}$$

Furthermore we have

$$(\Pi, \check{\Pi}) = (\bigcap_{i=0}^{\infty} \Pi_i, \bigcap_{i=0}^{\infty} \check{\Pi}_i)$$

and

$$(\Pi_{i+1}, \check{\Pi}_{i+1}) \succeq_{\text{GD}(\mathcal{E}, \Lambda)} (\Pi_i, \check{\Pi}_i), \quad i \geq 0.$$

Lemma A.3 Let $(\Pi_0, \check{\Pi}_0), (\Pi_1, \check{\Pi}_1), (\Pi_2, \check{\Pi}_2), \dots$ be a sequence of semantical structures as defined in Definition A.3. Let

$$F_i = \{\alpha \mid \Pi_i \models \alpha\} \text{ and } S_i = \{\beta, \omega \mid \check{\Pi}_i \models \beta, \omega\}$$

the set of Π_i -valid and $\check{\Pi}_i$ -valid, respectively, first order formulas.

We have

$$\begin{aligned} F_0 &= \text{Th}(\text{Form}(\mathcal{W})) \\ F_{i+1} &= \text{Th}(F_i \cup \{\omega_{i+1}\}) \end{aligned}$$

and

$$\begin{aligned} S_0 &= \text{Th}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W})) \\ S_{i+1} &= \text{Th}(F_i \cup \{\beta_{i+1}, \omega_{i+1}\}). \end{aligned}$$

Let $F = \bigcup_{i=0}^{\infty} F_i$ and $S = \bigcup_{i=0}^{\infty} S_i$, then we have

$$\Pi = \{\pi \mid \pi \models F\} \text{ and } \check{\Pi} = \{\pi \mid \pi \models S\}$$

and accordingly

$$F = \{\alpha \mid \Pi \models \alpha\} \text{ and } S = \{\beta, \omega \mid \check{\Pi} \models \beta, \omega\}.$$

Proof A.3 Trivial.

Thereby the letter F should remind us that we are dealing with the set of asserted formulas. Whereas S should refer to their supports.

Now, we can prove the following completeness theorem.

Theorem A.2 (Completeness)

Let $\Lambda = (D, \mathcal{W})$. If $(\Pi, \check{\Pi})$ is a \succeq_D -maximal element such that

$$\begin{aligned} (\Pi, \check{\Pi}) &\succeq_D \\ (\text{MOD}(\text{Form}(\mathcal{W})), \text{MOD}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W}))) \end{aligned}$$

then there is an extension \mathcal{E} of Λ such that

$$\Pi \models \text{Form}(\mathcal{E}) \text{ and } \check{\Pi} \models \text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E}).$$

Using Definition A.3 and Lemma A.3 it is enough to show that there exists an assertional extension \mathcal{E} of Λ , such that F corresponds to the asserted formulas of \mathcal{E} and S to their supports, i.e.

$$F = \text{Form}(\mathcal{E}) \text{ and } S = \text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E}).$$

According to [Brewka, 1991, Proposition 1] we have that

$$\mathcal{E} = \bigcup_{i=0}^{\infty} \mathcal{E}_i$$

is an assertional extension, if each \mathcal{E}_i is defined as follows

$$\mathcal{E}_0 = \mathcal{W}$$

and for each $i \geq 0$

$$\begin{aligned} \mathcal{E}_{i+1} &= \widehat{\text{Th}}(\mathcal{E}_i) \cup \{(\omega, \text{Supp}(\alpha) \cup \{\beta\} \cup \{\omega\}) \mid \\ &\quad \frac{\alpha : \beta}{\omega} \in D, \\ &\quad (\alpha, \text{Supp}(\alpha)) \in \mathcal{E}_i, \\ &\quad F \cup S \cup \{\beta\} \cup \{\omega\} \not\models \perp\} \end{aligned}$$

Thereby, it is enough to regard the following two cases.

$$1. \text{ Form}(\bigcup_{i=0}^{\infty} \mathcal{E}_i) = \text{Form}(\mathcal{E}) \subseteq F = \bigcup_{i=0}^{\infty} F_i,$$

$$\begin{aligned} \text{Form}(\bigcup_{i=0}^{\infty} \mathcal{E}_i) \cup \text{Supp}(\bigcup_{i=0}^{\infty} \mathcal{E}_i) \\ = \text{Supp}(\mathcal{E}) \cup \text{Form}(\mathcal{E}) \subseteq S = \bigcup_{i=0}^{\infty} S_i. \end{aligned}$$

Proof by induction over i .

$i = 0$ i. We have

$$\begin{aligned} \text{Form}(\mathcal{E}_0) &= \text{Form}(\mathcal{W}), \\ &\subseteq \text{Th}(\text{Form}(\mathcal{W})), \\ &= F_0, \\ &\subseteq F. \end{aligned}$$

ii. We have

$$\begin{aligned} \text{Form}(\mathcal{E}_0) \cup \text{Supp}(\mathcal{E}_0) &= \\ \text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W}), \\ &\subseteq \text{Th}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W})), \\ &= S_0, \\ &\subseteq S. \end{aligned}$$

$i \sim i+1$ Let $\text{Form}(\mathcal{E}_i) \subseteq F$ and $\text{Form}(\mathcal{E}_i) \cup \text{Supp}(\mathcal{E}_i) \subseteq S$. Regard $(\omega, \text{Supp}(\omega)) \in \mathcal{E}_{i+1}$.

i. With $(\omega, \text{Supp}(\omega)) \in \widehat{\text{Th}}(\mathcal{E}_i)$ we have $\omega \in \text{Th}(\text{Form}(\mathcal{E}_i)) \subseteq F$ as well as

$$\text{Supp}(\omega) \subseteq \text{Supp}(\mathcal{E}_i) \subseteq S.$$

ii. Otherwise, there is a default rule

$$\delta = \frac{\alpha : \beta}{\omega} \in D$$

with

$$(\alpha, \text{Supp}(\alpha)) \in \mathcal{E}_i$$

and

$$F \cup S \cup \{\beta\} \cup \{\omega\} \not\models \perp.$$

Now, we have to show that $\omega \in F$ and $\text{Supp}(\omega) \subseteq S$ holds. With $\Pi \models F$ the former is equivalent to $\Pi \models \omega$. Whereas with $\check{\Pi} \models S$ the latter is equivalent to $\check{\Pi} \models \text{Supp}(\omega)$.

A. Now, $(\alpha, \text{Supp}(\alpha)) \in \mathcal{E}_i \subseteq \mathcal{E}$ implies $\alpha \in \text{Form}(\mathcal{E}_i) \subseteq F$ and $\text{Supp}(\alpha) \in \text{Supp}(\mathcal{E}_i) \subseteq S$.

Therefore

$$\Pi \models \alpha \tag{1}$$

and

$$\Pi \models \text{Supp}(\alpha). \tag{2}$$

B. With

$$\check{\Pi} = \{\pi \mid \pi \models F \cup S\}$$

and

$$F \cup S \cup \{\beta\} \cup \{\omega\} \not\models \perp$$

we obtain

$$\exists \pi \in \check{\Pi}. \pi \models \{\beta\} \cup \{\omega\}. \tag{3}$$

Now, having that $(\Pi, \check{\Pi})$ is \succeq_D -maximal as well as (1) and (3) we get

$\Pi \models \omega$ or $\omega \in F$ respectively, and with (2) also

$$\check{\Pi} \models \text{Supp}(\alpha) \cup \{\beta\} \cup \{\omega\}$$

or

$$\text{Supp}(\omega) \in S, \text{ respectively.}$$

$$2. \bigcup_{i=0}^{\infty} F_i = F \subseteq \text{Form}(\mathcal{E}) = \text{Form}(\bigcup_{i=0}^{\infty} \mathcal{E}_i)$$

$$\bigcup_{i=0}^{\infty} S_i = S \subseteq \text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E})$$

$$= \text{Form}(\bigcup_{i=0}^{\infty} \mathcal{E}_i) \cup \text{Supp}(\bigcup_{i=0}^{\infty} \mathcal{E}_i)$$

Proof by induction over i .

$i = 0$ i. We have

$$\begin{aligned} F_0 &= \text{Th}(\text{Form}(\mathcal{W})), \\ &\subseteq \text{Form}(\mathcal{E}_1), \\ &\subseteq \text{Form}(\mathcal{E}). \end{aligned}$$

ii. We have

$$\begin{aligned} S_0 &= \text{Th}(\text{Form}(\mathcal{W}) \cup \text{Supp}(\mathcal{W})), \\ &\subseteq \text{Supp}(\mathcal{E}_1), \\ &\subseteq \text{Supp}(\mathcal{E}). \end{aligned}$$

$i \sim i+1$ According to the premises we have $F_i \subseteq \text{Form}(\mathcal{E})$ and $S_i \subseteq \text{Form}(\mathcal{E}) \cup \text{Supp}(\mathcal{E})$. Consider F_{i+1} and S_{i+1} . With $\delta_{i+1} \in D'$ we have

$$(\Pi_{i+1}, \check{\Pi}_{i+1}) \succeq_{\delta_{i+1}} (\Pi_i, \check{\Pi}_i) \quad (4)$$

as well as

$$F_{i+1} = \text{Th}(F_i \cup \{\omega_{i+1}\})$$

and

$$S_{i+1} = \text{Th}(S_i \cup \{\beta_{i+1}\} \cup \{\omega_{i+1}\}).$$

Now, we have to show that

$$(\omega_{i+1}, \text{Supp}(\alpha_{i+1}) \cup \{\beta_{i+1}\} \cup \{\omega_{i+1}\}) \in \mathcal{E}$$

holds.⁹

According to (4) we have $\Pi_i \models \alpha_{i+1}$ or $\alpha_{i+1} \in F_i \subseteq \text{Form}(\mathcal{E})$, respectively.

This implies that there exists a $k \in \{1, 2, \dots\}$ such that

$$(\alpha_{i+1}, \text{Supp}(\alpha_{i+1})) \in \mathcal{E}_k.$$

Since $\check{\Pi} \models F \cup S$ and $\{\beta_{i+1}\} \cup \{\omega_{i+1}\} \subseteq F \cup S$ we obtain

$$F \cup S \cup \{\beta_{i+1}\} \cup \{\omega_{i+1}\} \not\models \perp$$

Altogether this implies

$$\begin{aligned} (\omega_{i+1}, \text{Supp}(\alpha_{i+1}) \cup \{\beta_{i+1}\} \cup \{\omega_{i+1}\}) \\ \in \mathcal{E}_{k+1} \subseteq \mathcal{E} \end{aligned}$$

for a $k \in \{1, 2, \dots\}$.

Acknowledgements

Thanks to Wolfgang Bibel, Uwe Egly, Steffen Hölldobler, Jörg Schierstein, Josef Schneeberger and especially Gerd Brewka who fixed a bug in an earlier version of the paper. Thanks also to Susanne Schaub for improving my english style.

This research was supported by the Federal Ministry for Research and Technology within the project TASSO under grant no. ITW 8900 A7.

References

[Brewka, 1991] G. Brewka. Cumulative default logic: In defense of nonmonotonic inference rules. To appear in *Artificial Intelligence*, 1991.

⁹With $(\omega_{i+1}, \text{Supp}(\alpha_{i+1}) \cup \{\beta_{i+1}\} \cup \{\omega_{i+1}\}) \in \mathcal{E}$ we have also shown that $\omega_{i+1} \in \text{Form}(\mathcal{E})$ as well as that $\text{Supp}(\alpha_{i+1}) \cup \{\beta_{i+1}\} \cup \{\omega_{i+1}\} \subseteq \text{Supp}(\mathcal{E})$ holds.

[de Guerreiro and Casanova, 1990] R. A. de Guerreiro and M. A. Casanova. An Alternative Semantics for Default Logic. In K. Konolige, editor, *Third International Workshop on Nonmonotonic Reasoning, South Lake Tahoe, CA*, pages 141–157, 1990.

[Etherington, 1986] D. W. Etherington. *Reasoning with Incomplete Information: Investigations of Non-Monotonic Reasoning*. PhD thesis, Department of Computer Science, University of British Columbia, Vancouver, BC, 1986. Revised Version appeared as: Research Notes in AI, Pitman.

[Etherington, 1987] D. W. Etherington. A Semantics for Default Logic. In *Proceedings IJCAI-87, Milan, Italy*, pages 495–498, 1987.

[Etherington, 1988] D. W. Etherington. *Reasoning with Incomplete Information*. Research Notes in Artificial Intelligence. Pitman / Morgan Kaufmann, London, 1988.

[Lukaszewicz, 1988] W. Lukaszewicz. Considerations on Default Logic — An Alternative Approach. *Computational Intelligence*, 4:1–16, 1988.

[Makinson and Brewka, 1991] D. Makinson and G. Brewka. A Cumulative Inference Relation for JTMS and Logic Programming. In K. P. Jantke and P. H. Schmitt, editors, *Proceedings of the First Workshop on Nonmonotonic and Inductive Logic*, Lecture Notes in Artificial Intelligence. Springer Verlag, 1991. To appear.

[Makinson, 1989] D. Makinson. General Theory of Cumulative Inference. In M. Reinfrank, editor, *Proceedings of the Second International Workshop on Non-Monotonic Reasoning*, volume 346 of *Lecture Notes in Artificial Intelligence*, pages 1–18. Springer Verlag, 1989.

[McCarthy, 1980] J. McCarthy. Circumscription — A Form of Nonmonotonic Reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[Poole, 1989] D. L. Poole. What the Lottery Paradox tells us about Default Reasoning. In R. Brachman, H. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on the Principles of Knowledge Representation and Reasoning, Toronto*, pages 333–340, Los Altos, CA, May 1989. Morgan Kaufmann Publishers Inc.

[Reiter, 1980] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Reiter, 1987] R. Reiter. Nonmonotonic reasoning. *Annual Review of Computer Science*, 2:147–187, 1987.

[Schaub, 1991] T. Schaub. Assertional Default Theories: A Semantical View. Technical Report AIDA-91-1, FG Intellektik, FB Informatik, TH Darmstadt, Alexanderstraße 10, D-6100 Darmstadt, January 1991.

A Metalinguistic Treatment of Epistemic Contexts

Paul Schweizer
 Centre for Cognitive Science
 University of Edinburgh
 Edinburgh EH8 9LW
 Scotland

Abstract

Within the framework of a symbol processing model of knowledge and belief, it is natural to represent epistemic contexts as metalinguistic predicates attaching to names of expressions. However, theorems in [Montague, 1963] indicate that such an approach will be inconsistent, if the resulting epistemic logic is even as strong as the modal system $S1$. The present paper addresses this potential inconsistency, by exploring an approach in which the axioms of the epistemic logic are specified in terms of an iterated hierarchy of quotation names, defined on the basis of a standard first-order language. In the idealized limit case for the symbol processing account, the epistemic predicate is interpreted as attributing the property of deducibility, and this interpretation yields a consistent system of epistemic logic equivalent to propositional $S4$.

1 INTRODUCTION

Within the framework of a symbol processing model of knowledge and belief, it is natural to represent epistemic and doxastic contexts as predicates which attach to names of expressions, rather than as modal operators which attach to statements directly. This is because operators are interpreted as acting on the level of semantical content, rather than on the level of object language symbols. Thus under an explicitly syntactical rendition of epistemic contexts motivated by the symbol processing account, sentences qua notational entities are themselves designated by singular terms in the object language, and epistemic predicates then combine with these terms to form new atomic statements. If ϕ is a formula of the object language, $[\phi]$ the singular term denoting ϕ , and K is a predicate representing a given epistemic context, such as 'It is known that...', then the claim that ϕ is known will have

the form $K[\phi]$.¹ In this manner, assertions of knowledge and belief are represented as normal first-order predications, wherein a purely extensional property is attributed to a syntactical entity. Such an approach is intrinsically metalinguistic, since the formulas about which the epistemic claims are made must be interpreted as being *mentioned* rather than used.

Given this basic symbol processing framework, a human or machine 'agent' is represented as reasoning about its knowledge and beliefs through the manipulation of linguistic expressions, according to appropriate inferential transformations. This is a very natural and convenient method for computationally *modeling* basic epistemic processes, and this naturalness and convenience are quite independent both of metaphysical issues concerning the 'real' nature of abstract propositional content, and of psychological questions concerning the actual mechanisms whereby human beings process information. Indeed, the usefulness of the model simply for the purpose of computational simulation should be sharply distinguished from issues concerning metaphysical or psychological reification, and the concerns of the present paper are limited solely to the former domain.

Yet within this domain, there is a serious technical problem with regard to the formal coherence of the syntactical approach. Montague [1963] shows, via a direct extension of Tarski's theorem on the undefinability of truth, that if the modal concept of *necessity* is construed as a predicate of expressions, then the addition of even very weak modal principles will result in provable contradiction. Specifically, Montague shows that if the logic of the modal predicate N in the formal system L_N is governed by the law of necessity, together with the modal inference rule of necessitation, then inconsistency is imminent. Therefore, if the predicate modal system L_N satisfies the two general schemas:

¹In practice it may be desirable to index K to a particular agent, but such distinctions are not relevant to the following general discussion

- (i) $N[\phi] \rightarrow \phi$
- (ii) $\vdash N[\phi]$, if $\vdash \phi$

and if in addition, L_N possess the ‘self-referential’ power of elementary arithmetic, then there will be a closed sentence $\theta \in L_N$ such that $\vdash N[\theta] \wedge \neg N[\theta]$.

While Montague’s results are expressed in terms of the modal concept of necessity, it is quite clear that they can be easily extended to syntactical treatments of knowledge representation. According to classical accounts, something cannot be known unless it is true, and this principle is axiomatized by the schema

- (i') $K[\phi] \rightarrow \phi$.

Similarly, if it is assumed that one’s inference rules are sound, then if it is the case that one has a proof of ϕ , it should follow that one knows that ϕ . This is because the proof itself supplies adequate evidential justification for the knowledge claim. Such a principle can be formalized by the schema

- (ii') $\vdash K[\phi]$, if $\vdash \phi$,

which can be applied in a proof whenever the agent already has a proof of ϕ . All that is needed now is that the formal language possess sufficient power to sustain Gödel’s diagonal lemma, and an epistemic contradiction of the form $K[\beta] \wedge \neg K[\beta]$ will be provable in the system. The details of such a proof will be given in section 2 below.²

However, following ideas set forth in [Skyrms, 1978], the present paper will develop a very straightforward method for constructing consistent systems in which knowledge is interpreted as a first order predicate attaching to names of expressions, and in which (i'), (ii') and the diagonal lemma are all satisfied. The basic strategy is simply to separate two distinct features which are assumed to coalesce in Montague’s demonstration. Montague uses Gödel numerals as the names of formulas, and it is precisely with respect to these structurally rich terms that the diagonal lemma obtains. In the system L_K , which will be developed in section 3, inconsistency will be averted through the use of structurally primitive *quotation terms* as the privileged constants denoting syntactical objects. These primitive constants will be used to construct a metalinguistic hierarchy in which the levels are rigorously stratified, and in which the diagonal lemma is unable to subvert the crucial distinction between levels.

In accordance with the basic symbol processing paradigm, in which reasoning is modeled as the rule

²If use and mention are systematically conflated, so that a formula is construed as *denoting* itself, rather than as *asserting something about* itself, then this danger may not arise. Although this conflation is present in some of the AI literature, it constitutes a confusion rather than a solution.

governed manipulation of linguistic expressions, the knowledge predicate will be interpreted as attributing the property of deducibility. Thus the locution $K[\phi]$ will be construed as asserting that the formula ϕ is deducible in L_K . The construction will begin with an ordinary first-order language L_o , which lacks both quotation names and the epistemic predicate. The language L_1 is obtained by adding K along with singular terms for all closed formulas of L_o . An infinite hierarchy is generated by the fact that knowledge claims can be embedded to arbitrary finite depths, and consequently each iteration of the epistemic predicate will force an ascent to the next level in the hierarchy, which will possess a quotation name for each closed formula of the previous level. The union of all finite levels in the hierarchy will yield the fixed point L_K , which will then be able to model every embedded knowledge claim of arbitrary finite depth. The resulting system of epistemic logic will be shown to be equivalent to the modal system of propositional S4. Hence the epistemic predicate, when interpreted as provability, is more than sufficiently powerful to capture any plausible logic of knowledge or belief, which demonstrates that no weakening of the representational structure need be incurred by this metalinguistic approach.

If the epistemic predicate is interpreted as attributing the property of validity with respect to a class of models, rather than provability in a formal system, then a theory equivalent to propositional S5 can be obtained.³ Though this particular result will not be explored in the present paper, this stronger construction shows that even the most powerful systems of modal logic can be captured without loss within a purely first-order setting. Therefore, the contradiction derived by Montague is not caused by any fundamental defect of the predicate interpretation, but rather is due to the fact that the syntactical approach to modality and propositional attitudes results in a vast increase in expressive capacity. Within the predicate system explored by Montague, it is possible to construct and prove various ‘ungrounded’ modal formulas, which cannot in principle be articulated with the resources of the standard operator approach. And it is precisely these new modal formulas which lead to contradiction.

In order to correct for this, the strategy which will be employed in L_K is to readjust the delicate balance between expressive and deductive power, by defining the modal axioms of the system in terms of a rigidly stratified hierarchy. This will have the effect of rendering certain of the new formulas of L_K unprovable, since they will no longer count as instantiations of the axiom schemas. So consistency will be maintained by articulating the modal principles in a form which corre-

³This general result is discussed in Schweizer, ‘A Syntactical Approach to Modality,’ forthcoming in the *Journal of Philosophical Logic*

sponds exactly with the expressive power of the object level operator, and by this means, formal coherence is not undermined by the new modal formulas present in L_K .

2 THE ROLE OF THE DIAGONAL LEMMA

As mentioned above, Montague's proof relies upon the assumption that the modal predicate system in question is able to sustain Gödel's diagonal lemma. The diagonal lemma asserts that for every property $\psi(x)$ expressible in a given language L via a (possibly complex) formula with one free variable, it is provable that there exists a closed formula $\theta \in L$ such that

$$\vdash_L \theta \leftrightarrow \psi([\theta])$$

where $[\theta]$ is the Gödel numeral for θ . The diagonal lemma obtains in any system in which all recursive functions are representable, and thus it holds in any extension of the theory Q of Robinson arithmetic. So this 'self-referential' prerequisite of Montague's proof will be satisfied if it assumed that the system L_K of epistemic logic is an extension of Q .

Given this assumption, and the fact that the property of 'being known' is expressible in L_K via a formula with one free variable, namely $K(x)$, it follows by the diagonal lemma that there will be a sentence θ of L_K such that $\vdash \theta \leftrightarrow K[\theta]$. Thus the formula θ is true if and only if it is true that θ is known, and therefore θ is the epistemic analogue of the benign 'truth teller,' which asserts its own truth, and of the Henkin sentence which asserts its own provability.

Since the property of *not* being known is also expressible in L_K via a formula with one free variable, namely $\neg K(x)$, it follows that there will be another sentence $\mu \in L_K$ such that

$$(iii') \vdash \mu \leftrightarrow \neg K[\mu].$$

But unlike θ , μ is potentially malignant, and seems to yields a contradiction with the aid of schemas (i') and (ii'). The proof is as follows. $\vdash K[\mu] \rightarrow \neg \mu$ is entailed by (iii'), while $\vdash K[\mu] \rightarrow \mu$ is the instantiation of (i') with respect to μ . Together these yield $\vdash \neg K[\mu]$, which is one half of the contradiction. Then, $\vdash \neg K[\mu]$ yields $\vdash \mu$, by taking (iii') from right to left, and the provability of μ yields $\vdash K[\mu]$, by the application of (ii'). Therefore both $K[\mu]$ and $\neg K[\mu]$ appear to be deducible in L_K .

So the foregoing demonstration would seem to indicate that the system L_K must be inconsistent, if it is an extension of Q , and if the formal behavior of the predicate K is governed by the schemas (i') and (ii'), which are syntactical versions of relatively weak modal principles. How then can S4 be recovered from a pred-

icate approach to knowledge which contains Q ? The crucial point to notice is that in order for the above proof to go through, the object language terms denoting expressions in (i') and (ii'), *viz* terms of the form $[\phi]$, must be assumed to be Gödel numerals. Given a system such as L_K which contains arithmetic, and given the well developed methods for encoding syntax arithmetically, the inclination to use Gödel numerals to name formulas is quite natural. But it is not *compulsory* to do so. And it is precisely this assumption that the special axioms governing the behavior of K are expressed numerically which allows the diagonal lemma and the epistemic logic to syntactically mesh and generate a contradiction.

Hence in L_K , formal quotation terms will be used as the canonical names for sentences, and this feature of the system will be explicitly enshrined in the epistemic axioms. If ϕ is a sentence of L_K , then ' ϕ ' will be the standard name for ϕ . Furthermore, to eliminate notational ambiguity between different types of terms, the principles governing the logical behavior of K will have the explicit form:

- (i*) $K(' \phi ') \rightarrow \phi$
- (ii*) $\vdash K(' \phi '), \text{ if } \vdash \phi$

This syntactical gulf between the notational form of (iii') and that of the epistemic principles (i*) and (ii*), is sufficient to preserve the desired stratification of levels in the hierarchy, and to ensure that all knowledge claims provable in L_K through use of the modal axioms are well grounded. The 'self-referential' significance of the diagonal lemma with respect to knowledge claims will thereby be neutralized, because the diagonal lemma obtains only with respect to numerals, while the epistemic axioms are expressed via canonical quotation terms.

Quotation terms are not descriptive names with respect to the syntactical machinery of L_K , and in fact possess no internal structure accessible to the object level resources of the system. Thus quotation terms are too weak to sustain a diagonal lemma. For a given open formula $\psi(x) \in L_K$, there simply is no quotational fixed point θ such that $\vdash \theta \leftrightarrow \psi(' \theta ')$. Quotation constants lack the accessible internal content required to state and prove identities between different names, and hence cannot be used to construct the requisite 'definite descriptions' in the object language. The absence of quotational fixed points is especially blatant in L_K , since this particular language lacks standard names for open formulas. Quotation terms are supplied only for the sentences of L_K , and thus the relevant substitution operations on open formulas cannot be grammatically expressed using these terms.

However, since L_K is an extension of Q , the diagonal lemma will obtain for any given arithmetization of syntax. Thus there will be some sentence $\mu \in L_K$

such that $\vdash \mu \leftrightarrow \neg K[\mu]$. But such a sentence is a consequence of arithmetic, not modal logic, and its epistemic significance is radically curtailed by the fact that schemas (i*) and (ii*) are expressed in different terms. The inherent syntactical distinction between $[\mu]$ and ' μ ' prevents the diagonal lemma and the modal principles from formally interacting, since $K[\mu]$ and $K(' \mu ')$ remain notationally distinct. For example, nothing follows from $K[\mu] \rightarrow \neg \mu$ and the instance $K(' \mu ') \rightarrow \mu$ of principle (i*), since the antecedents are not the same, while $\vdash K(' \mu ')$ rather than $\vdash K[\mu]$ follows from $\vdash \mu$ by principle (ii*).

3 THE SYSTEM L_K

Now that the basic strategy has been outlined, the structural details of the system L_K will be presented. To begin, let the base L_o be the language of first-order arithmetic, possibly supplemented with additional vocabulary. In order to establish an equivalence with propositional $S4$, L_o will be used to define both a standard modal operator system L_\square , as well as the corresponding epistemic system L_K . Relevant equivalences between L_\square and L_K will then be demonstrated to obtain through the use of a mapping $C : L_\square \rightarrow L_K$.

Let the formulas of L_\square be the smallest set such that:

1. if $\phi \in L_o$, then $\phi \in L_\square$
2. if ϕ is a closed formula of L_\square , then so is $\square \phi$
3. if $\phi, \beta \in L_\square$, then so are $\neg \phi, (\phi \rightarrow \beta)$, and $\forall x \phi$

Thus L_\square is a typical sentential operator language wherein necessity applies only to closed formulas. The epistemic predicate system L_K will be defined as the union of an inductive hierarchy of languages, starting with L_o as the base. Since this construction is only intended to deal with *propositional* attitudes, quantification into epistemic contexts will be treated as vacuous.

Let the language L_{n+1} be the smallest set such that

1. terms of L_{n+1} :
 - (a) if t is a term of L_n , then t is a term of L_{n+1}
 - (b) if ϕ is a closed formula of L_n , then ' ϕ ' is an individual constant of L_{n+1}
2. formulas of L_{n+1} :
 - (a) if t is a term of L_{n+1} , then $K(t)$ is a formula of L_{n+1}
 - (b) if ϕ is a formula of L_n , then ϕ is a formula of L_{n+1}
 - (c) if $\phi, \beta \in L_{n+1}$, then so are $\neg \phi, (\phi \rightarrow \beta)$ and $\forall x \phi$

Let $L_K = \bigcup L_n$, for $n \in \omega$

The mapping C will associate with each formula ϕ of L_\square a corresponding formula $C(\phi) = \pi \in L_K$, where π will be thought of as the 'epistemic counterpart' of ϕ in L_K .

Let the mapping C be such that:

1. if $\phi \in L_o$, then $C(\phi) = \phi$
2. if ϕ is $\square \beta$, then $C(\phi) = K('C(\beta)')$
3. if ϕ is $\neg \beta, (\beta \rightarrow \psi)$, or $\forall x \beta$, then $C(\phi) = \neg C(\beta), (C(\beta) \rightarrow C(\psi))$, or $\forall x C(\beta)$, respectively

Now that the syntax of L_K has been defined, it is time to consider the semantics. Under the assumed representational scheme, knowledge is acquired by manipulating syntactical expressions according to a set of transformation rules, starting with a prespecified set of formulas which constitute the knowledge base. Thus in the ideal case, which supplies a natural bound for consistency results, what is known corresponds to what is (in principle) deducible from the initial set of known formulas. This means that the extension of the predicate K must be defined with reference to the transformation rules of the system. The process will begin with a standard first-order model M_o for L_o , which will then be inductively expanded to give the extensions of the predicate K and the new constants ' ϕ '. The infinite union of this procedure over the finite ordinals will result in a cumulative model M_K for L_K .

A base model M_o is the usual pair $\langle D_o, I_o \rangle$, consisting of a domain D_o and an extension function I_o . The first step in the expansion is to define a set T_o of initially provable sentences. Here there is room for variation depending on the particular epistemic agent under consideration, but for the present idealized purposes, T_o will be required to include all theorems of first-order logic and arithmetic. To this can be added a stock of specialized non-logical axioms for world knowledge, with the requirement that T_o be closed under first-order deduction.

So it is at this stage of the construction that a number of variations are possible, depending on what sort of epistemic context is being represented. To yield a natural limit case for knowledge, it is currently assumed that the inductive operation which generates the successive models for increasing levels in the hierarchy is such that the set of theorems at each level is deductively closed under the rules of first-order logic. If an agent with particular inferential or resource limitations were being modeled, then a correspondingly weaker operation could be utilized, which would simulate various sorts of incompleteness which afflict non-ideal agents.

An epistemic model M_{n+1} of L_{n+1} , together with its associated valuation function V_{n+1} , is generated by

a model M_o of L_o according to the following procedure, where Δ_o is some arbitrary set of compatible base models (including M_o) such that $M'_o \models T_o$ for every $M'_o \in \Delta_o$. Each such model M'_o will possess an associated set of provable formulas T'_o , and a set of compatible models Δ'_o , and it may well be the case that not $M_o \in \Delta'_o$.

1. $D_{n+1} = D_n \cup \{\phi : \phi \in L_n^c\}$ where L_n^c = the set of closed formulas of L_n
2. T_{n+1} is the largest set of $\phi \in L_{n+1}^c$ such that
 - (a) if $\phi \in T_n$, then $\phi \in T_{n+1}$
 - (b) if $\phi \in T_n$, then $K(\phi) \in T_{n+1}$
 - (c) if $\phi \in T_{n+1}$, then $M'_{n+1} \models \phi$, for all models $M'_{n+1} \in \Delta_{n+1}$ (where Δ_{n+1} is the set of M'_{n+1} generated from Δ_o)
 - (d) T_{n+1} is closed under first-order logic plus the rule $\vdash K(\phi)$ if $\vdash \phi$, for all $\phi \in L_n^c$
3. $I_{n+1}(\phi) = \phi$, for all $\phi \in L_n^c$
4. $I_{n+1}(a) = I_n(a)$ for all constants a of L_n
5. $I_{n+1}(K) = \{\phi : \phi \in T_n\}$

A model M_K of L_K induced by a model M_o of $L_o = \bigcup M_n$, for $n \in \omega$

Clause 5. in the above definition supplies the basis for a standard first-order account of truth for epistemic predications. A sentence of the form $K(\phi)$ will be true in a model iff

$$I_K(\phi) \in I_K(K).$$

Since the models are cumulative and $I_K(\phi)$ must equal ϕ , the formula $K(\phi)$ will be true iff $\phi \in T_n$ for some level n in the hierarchy.

As will now be demonstrated, the class of models defined in this manner makes valid exactly the set of epistemic counterparts of theorems of propositional S4. In this sense, L_K constitutes a precise mirroring of the standard operator approach. The equivalence will be established through the following two theorems.

Theorem 1: If ϕ is a sentence of L_\square and is a theorem of propositional S4, then $C(\phi)$ is true in all models of L_K .

proof: The demonstration will first establish that the counterparts of the S4 axioms are true in all epistemic models, and then that the inference rules preserve validity.

1. $K(\phi \rightarrow \psi) \rightarrow (K(\phi) \rightarrow K(\psi))$. Suppose the antecedent is true in a model M_K . Let n be the level in the hierarchy at which the axiom first appears. According to clauses 4. and 5., $M_n \models K(\phi \rightarrow \psi)$ iff $\phi \rightarrow \psi \in T_{n-1}$. Suppose further that $M_K \models K(\phi)$. Then $\phi \in T_{n-1}$, in which case $\psi \in T_{n-1}$, since the set of theorems is

closed under modus ponens. Therefore $K(\psi)$ is true in M_n , and since the models are cumulative, $M_K \models K(\psi)$, which means that the axiom must be true in every epistemic model.

2. $K(\phi) \rightarrow \phi$. Suppose $M_K \models K(\phi)$, and that L_n is the language in which the axiom first occurs. By clauses 4. and 5., it must be the case that $\phi \in T_{n-1}$. By the soundness constraint of clause 2.(c) it must be the case that $M_{n-1} \models \phi$, and since the models are cumulative, $M_K \models \phi$.
3. $K(\phi) \rightarrow K(K(\phi))$. Suppose $M_K \models K(\phi)$, and that L_n is the language in which the axiom first occurs. Then $\phi \in T_{n-1}$, and by clause 2.(b), $K(\phi) \in T_n$, so that $K(K(\phi))$ is true in M_{n+1} , and since the models are cumulative, $M_K \models K(K(\phi))$.

Since quantification into epistemic contexts is treated as vacuous, only the inference rules of modus ponens and necessitation will be of concern. Consider the rule of necessitation, expressed as (iii*) $\vdash K(\phi)$, if $\vdash \phi$. Since K is to be interpreted as making assertions about provability, the fact that T_{n+1} is defined as the largest set satisfying the soundness constraint will be required to preserve the validity of iterated claims about provability. If $\phi \in L_{n+1}$ is valid, then it will be true in every model in Δ_{n+1} , and hence will be included in T_{n+1} . Thus, by clauses 4. and 5., $K(\phi)$ will be true in M_{n+2} , and necessitation will preserve validity. The case for modus ponens is straightforward •

Theorem 2: If $\phi \in L_\square$ is not a theorem of S4, then there is a model M_K of L_K such that $M_K \models C(\neg\phi)$.

proof: By giving a procedure for converting every S4 Kripke model into a corresponding epistemic predicate model, and then proving that for every sentence ϕ of L_\square , the truth value of ϕ in the Kripke model is identical to the truth value of $C(\phi)$ in the corresponding epistemic model.

An S4 Kripke model for the modal operator language L_\square is the triple $\langle G, W, R \rangle$, where W is a set of valuations over the atomic sentences of the language and R is a reflexive and transitive relation on W . Since only the propositional behavior of the modal device applied to closed formulas is of current interest, each first-order model M_o can be viewed as specifying a propositional valuation M_o^p , where all first-order atomic sentences, and all closed formulas with a quantifier as main connective, are treated as propositional variables. In this manner, the predicate structure of the sentences of L_o will be left unanalyzed, while the admissible valuations will nonetheless

less implicitly respect this structure. So only those propositional valuations M_o^p which are specified by a first order model M_o will be allowed as members of W .

The Kripke model $\langle G, W, R \rangle$ induces a valuation V^G over all the sentences of L_\square such that $V^G(\neg\phi) = 1 - V^G(\phi)$, $V^G(\phi \vee \psi) = V^G(\phi) + V^G(\psi)$, etc., for the truth functions, and such that $V^G(\square\phi) = 1$ if $V^X(\phi) = 1$, for all X such that $X \in W$ and GRX , and $V^G(\square\phi) = 0$ otherwise.

An epistemic model M_K^G is constructed from a given Kripke model $\langle G, W, R \rangle$ according to the following procedure. The first-order model M_o used to specify G will comprise the base. The set of theorems T_o is the set of all sentences $\phi \in L_o$ such that $V^G(\square\phi) = 1$, and the set Δ_o is defined as the collection of all first-order models which correspond to the valuations $X \in W$ such that GRX . Then the model $M_{n+1}^{G,W}$ of L_{n+1} is generated as above. So Kripke's method of giving a set of possible worlds within each model is recaptured in the metalinguistic approach by appeal to the set of alternate *models* contained in Δ . But it is important to note that the resulting epistemic model is first-order, since this appeal to alternate models is used to specify the extension of a (metalinguistic) predicate, rather than to define the truth conditions for modal formulas.

lemma: For all sentences $\phi \in L_\square$ and all Kripke models $\langle G, W, R \rangle$, it is the case that $V^G(\phi) = V_K^G(C(\phi))$, where V_K^G is the valuation induced by the epistemic model M_K^G corresponding to $\langle G, W, R \rangle$.

proof: By induction on the structural complexity of formulas of L_\square and the iterative construction of epistemic models.

1. If ϕ is a sentence of L_o , then $C(\phi) = \phi$, and therefore $V^{G,W}(\phi) = V_K^{G,W}(\phi) = 1$, if $M_o \models \phi$, and 0 if not $M_o \models \phi$.
2. Suppose ϕ is a sentence of L_\square of the form $\square\beta$ and $V^G(\phi) = 0$. Then there is some $X \in W$ such that GRX and $V^X(\beta) = 0$. There are two possible cases.
 - (a) Suppose $X = G$, and that L_n is the level of the hierarchy in which $C(\beta)$ first appears. By induction hypothesis, $V_n^G(C(\beta)) = 0$. But if $V_n^G(C(\beta)) = 0$, then $C(\beta)$ cannot be in T_n , since this would violate the soundness requirement of clause 2.(c). And if not $C(\beta) \in T_n$, then $V_{n+1}^G(K('C(\beta)')) = 0$, by clause 5., and since the models are cumulative, this entails that $V_K^G(K('C(\beta)')) = 0$. But $K('C(\beta)') = C(\square\beta) = C(\phi)$, and hence $V_K^G(C(\phi)) = V^G(\phi) = 0$.

(b) Suppose $X \neq G$. Take the epistemic model M_K^X generated on the basis of M_o^X . Since $V^K(\beta) = 0$, then by induction hypothesis, where n is the level in the hierarchy at which $C(\beta)$ first appears, it must be the case that $V_n^X(C(\beta)) = 0$. Since $M_n^X \in \Delta_n$ for the model M_n^G and $V_n^X(C(\beta)) = 0$, it follows that not $C(\beta) \in T_n^G$. But if not $C(\beta) \in T_n^G$, then $V_{n+1}^G(K('C(\beta)')) = 0$ by clause 5, and thus $V_K^G(C(\phi)) = 0$.

3. On the other hand, suppose that ϕ is a sentence of L_\square of the form $\square\beta$ and that $V^G(\phi) = 1$. Then $V^X(\beta) = 1$, for all X such that $X \in K$ and GRX . Again, let n be the level in the hierarchy at which $C(\beta)$ first occurs. By induction hypothesis, $V_n^X(C(\beta)) = 1$ for all $M_n^X \in \Delta_n$. Since T_n^G has been defined as the largest set which satisfies the constraints, it follows that $C(\beta) \in T_n^G$, which means that $V_{n+1}^G(K('C(\beta)')) = 1$, and hence that $V_K^G(C(\phi)) = V^G(\phi) = 1$.
4. The case for the truth functional connectives is straightforward •

It is perhaps useful to employ these epistemic models to provide a counterexample to a particular *S4* non-theorem. Suppose β is an atomic sentence of L_o , and hence that not $\beta \in T_o$ for some base model M_o . Let M'_o be in Δ_o and be such that $\beta \in T'_o$. Then $M_1 \models \neg K('\beta')$ while $M'_1 \models K(''\beta'')$. But $\neg K(''\beta'')$ is not in T_1 , because this formula is false in M'_1 . So $K(''\neg K(''\beta''))$ is false in M_2 , which means that the epistemic counterpart of the *S5* axiom $\neg \square\phi \rightarrow \square \neg \square\phi$ is false in M_K . Hence it can be true that a formula is unprovable, and false that this truth is itself provable.

It is also important to note the role of Gödel's second incompleteness theorem in this context. The law of necessity is an axiom of *S4* and is valid in L_K . Thus $\vdash K(''\beta'') \rightarrow \beta$, even when *not* $\vdash \beta$. But K is not the canonical proof predicate of Gödel's theorem, and K is not definable in arithmetic. Therefore no contradiction arises from this feature of L_K .

4 CONCLUSION

The system T_K shows that epistemic contexts of the form 'It is known that ...' can be consistently represented as first-order predicates of syntactical objects, if the expressive and deductive power of the system are calibrated to reflect the structure of the object level modal operator. This is accomplished by defining the epistemic logic in terms of a rigorously stratified hierarchy of quotation names, which are structurally incapable of diagonal self-reference. This approach yields results equivalent to the familiar operator versions of modal logic, because operators must attach directly to formulas, rather than to their names, and this precludes the possibility of constructing a modal

statement by attaching the operator to a term which denotes that formula itself. Thus iterated operator modality is inherently hierarchical, and this structural feature of the operator is reproduced in the present system through an iterated hierarchy of terms.

In the translation based approach to predicate modality examined in [Rivieres and Levesque, 1986], consistency is maintained not through the explicit use of a restrictive hierarchy of terms, but rather by embedding the operator language L_{\Box} in a first-order system with a 1-place predicate L for modality. The natural embedding, defined with respect to some class of encoding terms (such as Gödel numerals), specifies a proper subset of the modal predicate language, since expressions with open terms in the scope of L will not be in the range of the embedding. For example, the formula $L(x)$ does not correspond to any modal expression of L_{\Box} , since the operator must attach to a sentence, and a sentence must be encoded as a closed term; locutions such as $\Box(x)$ are obviously ill-formed. The range of the embedding thereby delimits a set of 'regular' formulas, and consistency is preserved by restricting attention to this regular subset of the language.

The present approach is related to the translation based construction through the fact that the encoding terms occurring in the regular formulas will implicitly define a well grounded hierarchy. Thus the same effect could be achieved by constructing an iterated hierarchy of privileged encoding terms within the modal predicate language, and using this hierarchy to define the set of regular formulas. There is a 1-1 correspondence between the quotation terms in the metalinguistic structure of L_K , and the encoding terms defined through a mapping based on L_{\Box} , because iterated operator modality is inherently stratified.

References

- Boolos, J. 1979. *The Unprovability of Consistency*. Cambridge University Press.
- Carnap, R. 1937. *The Logical Syntax of Language*. Routledge and Kegan Paul, Ltd., London.
- Gödel, K. 1933. 'An Interpretation of the Intuitionistic Sentential Calculus.' *Ergebnisse eines mathematischen Kolloquiums*, Vol. 4. Verlag Franz Deuticke, Vienna.
- Gupta, A. 1982. 'Truth and Paradox.' *Journal of Philosophical Logic* 11, 1-60.
- Konolige, K. 1988. 'Belief and Incompleteness.' Chapter 10 in *Formal Theories of the Common Sense World*, Norwood, N.J., Ablex.
- Kripke, S. 1959. 'A Completeness Theorem in Modal Logic,' *Journal of Symbolic Logic* 24 (1959) 1-15.
- Montague, R. 1963. 'Syntactical Treatments of Modal-

ity, with Corollaries on Reflexion Principles and Finite Axiomatizability.' Appearing in *Formal Philosophy*, R. Thomason (ed), Yale University Press, 1974.

Quine, W.V. 1953. 'Three Grades of Modal Involvement.' *Proceedings of the XIth International Congress of Philosophy*, Vol. 14. North-Holland Publishing Company.

Rivieres, J. and H. Levesque, 1986. 'The Consistency of Syntactical Treatments of Knowledge,' appearing in *Theoretical Aspects of Reasoning About Knowledge*, J. Halpern (ed.), Morgan Kaufman, Los Altos, California.

Russell, B. 1940. *An Inquiry into Meaning and Truth*. George Allen and Unwin, Ltd., London. See chapter IV.

Schweizer, P. 1987. 'Necessity Viewed as a Semantical Predicate.' *Philosophical Studies* 52, pp. 33-47.

- 1989. *A Metalinguistic Interpretation of Modality*. Doctoral Dissertation, University of California at Irvine.

Skyrms, B. 1978. 'An Immaculate Conception of Modality.' *Journal of Philosophy* 75, pp. 77-96.

Smorynski, C. 1985. *Self-Reference and Modal Logic*. Springer-Verlag, New York.

Smullyan, R.M. 1957. 'Languages in Which Self Reference is Possible.' *Journal of Symbolic Logic* Vol. 22, pp. 55-67.

Solovay, R. 1976. 'Provability Interpretations of Modal Logic.' *Israel Journal of Mathematics*.

The Completeness of DRAT, A Technique for Automatic Design of Satisfiability Procedures

Jeffrey Van Baalen
Price Waterhouse Technology Centre
68 Willow Road
Menlo Park, CA 94025

Abstract

It is a familiar idea to use specialized methods to accelerate a general problem solver. We have developed and implemented a technique that, given a problem stated in first-order logic, designs a procedure to decide a theory containing as many of the problem's axioms as possible. Such procedures are interfaced to theorem prover so that the theorem prover and decision procedure "work together" to solve the original problem. We present a formalization of this technique and prove that it is *complete*, i.e., that any problem solvable by the theorem prover can also be solved by the decision procedure/theorem prover combination. Usually these combinations solve problems much more efficiently than the theorem prover alone.

1 INTRODUCTION

It is a familiar idea to use specialized methods to accelerate a general problem solver. Perhaps the best examples of this are the specialized methods that have been developed to accelerate theorem provers. For instance, specialized treatments of equality to accelerate theorem proving are common[13][4, ch.5] and specialized reasoners have been added to theorem provers to accelerate taxonomic reasoning[1, 2]. Given a general problem solver with a collection of specialized methods, a problem stater can improve the efficiency with which a problem is solved by formulating it so that as much of the reasoning as possible is done by the specialized methods, avoiding the weaker general method. But this requires that the problem stater be intimately familiar with the specialized methods that the problem solver knows. When the problem is not formulated to take advantage of specialized methods, the general problem solver must be used, even though better methods are available.

Thus, the idea of using specialized methods to acceler-

ate general problem solving, while intuitively appealing, raises a difficult issue. Given a library of specialized methods and a problem stated without knowledge of those methods, how can the appropriate methods for solving the problem be identified automatically? A technique for doing this would separate the problem stating process from the problem solving process and still allow specialized methods to be taken advantage of. Unfortunately, the development of such techniques has received little study.

We have developed and implemented a technique called DRAT that, given a problem stated in first-order logic, can often produce a type of decision procedure for a significant subset of the problem. This type of decision procedure, called a *satisfiability* procedure, decides for a theory T whether or not a conjunction of ground literals is satisfiable in T . A satisfiability procedure produced by DRAT for a problem is interfaced with a general-purpose theorem prover in such a way that the procedure and the theorem prover work together to solve the problem. This satisfiability procedure can also be used along with the theorem prover to solve other problems in the same *class* as the input problem.

In practice, the resulting problem solvers have been much more efficient than the theorem prover solving the problem alone because many of the inferences that the theorem prover would have to do are done more efficiently by the satisfiability procedure.

Let Ψ be the set of axioms of a problem and let S be the satisfiability procedure that DRAT designs for Ψ' , some subset of Ψ . The theorem prover restricts its manipulation of the statements in Ψ' , using S instead whenever possible. This paper presents a formalization of DRAT and proves that it is complete, i.e., that for any first-order statement ϕ , if $\Psi \models \phi$, S combined with the theorem prover will prove ϕ . We present only a brief description of the DRAT algorithm here. A detailed description of an implementation can be found in [11] or [12].

DRAT was inspired by human problem solving perfor-

mance on analytical tasks of the type found on graduate level standardized admissions tests. An example problem is given in Figure 1.

Given: M, N, O, P, Q, R, and S are all members of the same family. N is married to P. S is a grandchild of Q. O is a niece of M. The mother of S is the only sister of M. R is Q's only child. M has no brothers. N is a grandfather of O.

Query: Who are the siblings of S?

Figure 1: The FAMILIES Analytical Reasoning Problem

We analyzed human problem-solving behavior on a number of these problems and found the prevalent use of diagrams to assist in problem solving. Figure 2 illustrates the typical diagrams people use to solve the problem in Figure 1.



"R is the only child of Q" "S is a grandchild of Q"
 (Divided rectangles represent couples; circles represent sets of children of the same couple: full circles are closed sets, broken circles are sets all of whose members may not be known; the directed arc represents the "children-of" function between couples and their sets of children.)

Figure 2: Two statements in a representation commonly used by people.

These diagrams were found to contain a common set of structures (across different people and different problems). The arcs in Figure 2 are an example of such a structure. They represent the 1-1 function between a married couple and their set of children. Each common structure was also found to have a standard set of procedures for manipulating it. For example, one procedure associated with the arcs in Figure 2 ensures that they behave like a 1-1 function. It reads roughly as, "If two objects are equal and they appear at the same end of two separate 1-1 function arcs with the same function symbol, the arcs and the objects at their other end can be composed." This procedure is among those used to compose the structures in Figure 2 to yield the diagram in Figure 3.

People use these diagrams to test the satisfiability of a particular collection of facts by creating the structures representing each fact and then composing them. The conjunction is satisfiable just in case no contradiction is signalled in the composition process.

DRAT has a library of procedures called *schemes*. These schemes model people's diagrammatic struc-

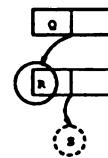


Figure 3: Composition of the structures in Figure 3.

tures and their manipulations. Schemes were found to have a number of important properties which are described in this paper. Perhaps the most important of these properties is that each scheme turns out to be a satisfiability procedure. Another important property of schemes is that they can be used as building blocks to construct "larger" satisfiability procedures. DRAT uses this property to construct satisfiability procedures for input problems.

The implementation of DRAT includes the schemes found in analyzing the diagrams that people used on thirty analytical tasks. It has been tested on twelve of these problems stated in a sorted first-order logic. The problems vary in size from thirty to sixty sorted first-order statements. The performance of the theorem prover/satisfiability procedure combinations that DRAT produces for these problems was at least two orders of magnitude better than the performance of the theorem prover alone. For example, our general theorem prover took 988,442 resolutions—three hours and five minutes—to solve the problem shown in Figure 1. The satisfiability procedure that DRAT produced was able to solve the problem entirely without the theorem prover and did so in less than three seconds.

2 PRELIMINARIES

Each scheme is a *tractable literal satisfiability procedure* for a *theory*.

Definition 1 A *theory* is a set of statements in first-order predicate calculus with equality.

Definition 2 A *literal satisfiability procedure* for a theory T is a procedure that decides for any conjunction of ground literals Σ whether or not $\Sigma \cup T$ is satisfiable.

Each scheme is *tractable* in the sense that, given any Σ containing n literals, the scheme for a theory T decides the satisfiability of $\Sigma \cup T$ in time polynomial in n .

Given a particular Σ , in addition to determining literal satisfiability in some theory T , each scheme computes $\{u = v \mid u, v \in C \wedge \Sigma \cup T \models u = v\}$, where C is the set of constant symbols appearing in Σ . As detailed in section 3, these equalities are communicated between schemes in a way that allows the combination of schemes to determine satisfiability for the union of their theories.

One important result of this research is the particular library of schemes we have developed from the observation of human problem solving of analytical tasks. However, in the formal characterization that follows, we abstract away from the detail of the current scheme library, identifying the properties of schemes required for the completeness of DRAT.

This paper first takes a simplified view of what DRAT will accept as an input problem and also assumes that DRAT is only successful if it can produce a satisfiability procedure for an entire problem. In this setting, we prove that a combination of schemes is a satisfiability procedure for the union of the theories of the individual schemes. In section 7, the above restrictions are relaxed and it is shown how, in the more general setting, the procedures produced by DRAT are interfaced with a theorem prover.

DRAT requires that the formulas of schemes and the formulas of an input problem be converted to *clauses*, i.e., disjunctions of first-order literals. The remainder of the paper assumes that this has been done. However, the presentation will often use more intuitive forms for statements, when the conversion to clause form is straightforward.

The restricted definition of a problem taken first is:

Definition 3 A *problem* is a triple $\langle \Sigma, T_C, \Phi \rangle$, where Σ and Φ are sets of ground literals and T_C is a set of clauses each of which contains at least one variable. Such a triple is interpreted as a question about whether or not for all the ground literals $\phi \in \Phi$, $\Sigma \cup T_C \models \phi$.

Here is an example problem:

$$\begin{aligned} \Sigma_1 &= \left\{ \begin{array}{l} \text{grandfather}(O, N), \text{married}(N, P) \\ \text{grandchild}(S, Q), \text{niece}(O, M), \\ M \neq N, N \neq O, \dots \end{array} \right\} \\ T_{C_1} &= \left\{ \begin{array}{l} \text{mother}(S, x) \leftrightarrow \text{sister}(M, x), \\ (\text{sister}(M, x) \wedge \text{sister}(M, y)) \Rightarrow x = y, \\ \text{child}(Q, x) \leftrightarrow x = R, \\ \neg\text{brother}(M, x), \dots \end{array} \right\} \\ \Phi_1 &= \{\text{sibling}(O, S), \text{child}(N, M)\} \end{aligned}$$

In addition to those axioms shown, Σ_1 also contains disequalities between all of the individual constants mentioned. T_{C_1} also contains definitions of concepts such a *grandchild* and formulas defining general properties of the family relation domain such as symmetry of *married*.

Given a problem $\langle \Sigma, T_C, \Phi \rangle$, DRAT's objective is to design a literal satisfiability procedure for T_C . This procedure is used to solve the problem for the particular Σ and Φ . To determine whether for some $\phi \in \Phi$, $\Sigma \cup T_C \models \phi$, the satisfiability procedure for T_C is used to decide whether or not $\Sigma \cup T_C \cup \neg\phi$ is unsatisfiable. For example, DRAT tries to design a satisfiability procedure for T_{C_1} . If successful, the procedure is used

to decide whether "O" is a sibling of "S" and "M" is a child of "N" follow from $\Sigma_1 \cup T_{C_1}$ by determining the satisfiability of $\Sigma_1 \cup T_{C_1} \cup \neg\text{sibling}(O, S)$ and of $\Sigma_1 \cup T_{C_1} \cup \neg\text{child}(N, M)$.

Obviously, we are better off using a satisfiability procedure for T_C to solve a problem $\langle \Sigma, T_C, \Phi \rangle$ than using a general theorem prover because the satisfiability procedure is guaranteed to halt. Perhaps less obvious is the fact that these procedures are usually much more efficient than a general theorem prover. The intuition behind this is that the complexity of the theorem prover solving the problem is a function of the size of the entire problem, while the complexity of the satisfiability procedure is a function of the size of $\Sigma \cup \Phi$. As pointed out in section 1, this intuition is substantiated by the performance of the procedures that DRAT has designed.

3 THE DRAT TECHNIQUE

We will call the relation, function and individual constant symbols in a theory the *nonlogical symbols* of that theory. The nonlogical symbols of each scheme's theory are treated as parameters to be instantiated with the nonlogical symbols of T_C . For example, the scheme $T_{\text{symmetric}}$ whose theory is $\{R(x, y) \Rightarrow R(y, x)\}$ is parameterized by R .

DRAT tries to find a set of scheme instances that can be combined to give a literal satisfiability procedure for T_C . Consider a set of scheme instances. Call the union of the theories of each scheme instance T_I . DRAT has succeeded in finding a satisfiability procedure when it finds a T_I that is logically equivalent to T_C . The following is an abstract description of this process:

```

instances ← ∅
T_I ← ∅
T'_C ← T_C
UNTIL empty(T'_C) DO
    instance ← choose-instance(T'_C)
    IF null(instance) THEN EXIT-WITH failure
    instances ← union(instance, instances)
    T_I ← union(theory(instance), T_I)
    FOR EACH φ ∈ T'_C
        WHEN T_I ⊨ φ DO T'_C ← T'_C - φ
    END FOR
END UNTIL

```

A set of scheme instances is built up incrementally and, simultaneously, the set of clauses in T'_C is paired down. Each time *choose-instance* is invoked, it inspects T'_C and chooses a scheme instance whose theory is entailed by T'_C . After the theory of *instance* is added to T_I , DRAT removes clauses from T'_C that are entailed by T_I .

DRAT uses the following procedure for computing satisfiability in T_I to determine the $\phi \in T'_C$ that follow from T_I . For each clause ϕ , it creates ϕ' by substitut-

ing a new individual constant for each unique variable in ϕ . If the satisfiability procedure for T_I reports that $\neg\phi' \cup T_I$ is unsatisfiable, $T_I \models \phi$.

If the algorithm is exited with T'_C empty, DRAT has succeeded in finding a T_I that is equivalent to T_C . To see this, note that $T'_C \cup T_I \equiv T_C$ is an invariant of the loop. Adding theory(*instance*) to T_I does not violate this condition because $T'_C \models \text{theory(instance)}$. Removing from T'_C clauses ϕ such that $T_I \models \phi$ also does not violate the condition.

If the algorithm is exited because `choose-instance` returns nil, it has failed to find a T_I that is equivalent to T_C .

Note that this algorithm is nondeterministic because, in general, on a call to `choose-instance`, there are several instances from which to choose. The DRAT implementation searches for an appropriate collection of scheme instances. This search is reduced considerably by the fact that scheme instances in T_I may not share nonlogical symbols. As discussed in section 6, this restriction is required to allow schemes to be combined by the method described below. More detail on how the DRAT implementation controls this search can be found in [12].

3.1 A PROCEDURE FOR COMBINING SCHEMES

Since T_I is the theory of a set of scheme instances, so long as these instances do not share nonlogical symbols, DRAT has a satisfiability procedure for T_I . This procedure is the combination of schemes used to create T_I . DRAT's combination technique is the same technique as reported by Nelson & Oppen in [5] and a more detailed description than what follows can be found there.

Let $\mathcal{L}(T)$ be the set of nonlogical symbols appearing in the clauses of T . We will often refer to $\mathcal{L}(T)$ as the language of T . Consider two scheme instances, T_1 and T_2 , where $\mathcal{L}(T_1)$ is disjoint from $\mathcal{L}(T_2)$, and consider a conjunction of literals Σ in $\mathcal{L}(T_1 \cup T_2)$. The procedure for deciding the satisfiability of $\Sigma \cup T_1 \cup T_2$ begins by splitting Σ into two conjunctions of literals: Σ_1 , with literals in $\mathcal{L}(T_1)$ and Σ_2 , with literals in $\mathcal{L}(T_2)$ such that the conjunction of literals in Σ_1 and Σ_2 is satisfiable just in case Σ is.

When a literal in Σ contains nonlogical symbols from $\mathcal{L}(T_1 \cup T_2)$, remove each subterm whose function symbol is not in the language of the head symbol of the term. A subterm is removed by substituting a new constant symbol for that subterm in the literal and conjoining an equality between the term and the new symbol with the proper Σ_i . For example, suppose R is in $\mathcal{L}(T_1)$, f is in $\mathcal{L}(T_2)$ and Σ contains the literal $R(f(a))$. The embedded term is in the wrong language, so it is removed. This is done by substituting

a new constant, say b , for $f(a)$ in $R(f(a))$ to obtain $R(b)$ and conjoining $b = f(a)$ with Σ_2 .

For each literal in Σ , this technique is applied repeatedly to the right most function symbol in the wrong language until the literal no longer contains symbols in the wrong language. Then the literal is conjoined with the appropriate Σ_i . For instance, $R(b)$ from the example above contains no symbols in the wrong language so it is conjoined with Σ_1 .

Next the scheme for T_1 is used to determine the satisfiability of $\Sigma_1 \cup T_1$. Recall that in so doing, this scheme also computes the set of equalities between constants in Σ_1 that follow from $\Sigma_1 \cup T_1$. Call this set E_1 . The scheme for T_2 is used to determine the satisfiability of $\Sigma_2 \cup T_2 \cup E_1$. If it is satisfiable, E_2 , the set of equalities that follow from $\Sigma_2 \cup T_2 \cup E_1$, is propagated back to T_1 , i.e., T_1 is used to compute $\Sigma_1 \cup T_1 \cup E_2$.

This propagation of equalities continues until one of the schemes reports "unsatisfiable" or until no new equalities are computed. Note that since there are at most $n - 1$ nonredundant equalities between n constant symbols, this process will terminate. Unless the scheme for T_1 or T_2 reports "unsatisfiable," the procedure for the combination returns "satisfiable."

A complication to this equality propagation procedure is that given a set of ground literals, many tractable schemes imply disjunctions of equalities between constants without implying any of the disjuncts alone, a property called *nonconvexity* in [5]. An example of a convex scheme is one that determines satisfiability for the theory of equality with uninterpreted function symbols. An example of a nonconvex scheme is one for the theory of sets. To see this, note that $\{a, b\} = \{c, d\}$ implies $a = c \vee a = d$, but does not imply either equality alone.

A scheme associated with a nonconvex theory must compute disjunctions of equalities between constants that follow from a given conjunction of ground literals. The equality propagation procedure is extended to handle such schemes by case splitting when a non-convex scheme produces a disjunction. When one of the component schemes produces the disjunction $c_1 = d_1 \vee \dots \vee c_n = d_n$, the combined satisfiability procedure is applied recursively to the conjunctions $\Sigma_1 \cup \Sigma_2 \cup \{c_1 = d_1\}, \dots, \Sigma_1 \cup \Sigma_2 \cup \{c_n = d_n\}$. If any of these is satisfiable, "satisfiable" is returned, otherwise "unsatisfiable" is returned.

As a simple example of this procedure, consider two schemes: \mathcal{E} for the theory of equality with uninterpreted function symbols and \mathcal{S} for the theory of finite sets. Now consider whether

$$\Sigma = \left[\begin{array}{l} f(a) = \{b, g\} \wedge f(c) = \{d, e\} \wedge a = c \wedge \\ g \neq d \wedge g \neq e \wedge b \neq d \wedge b \neq e \end{array} \right]$$

is satisfiable. First Σ is split into

$$\Sigma_1 = \left[\begin{array}{l} a = c \wedge g \neq d \wedge g \neq e \wedge b \neq d \wedge b \neq e \wedge \\ f(a) = c_1 \wedge f(c) = c_2 \end{array} \right]$$

$$\Sigma_2 = [c_1 = \{b, g\} \wedge c_2 = \{d, e\}].$$

\mathcal{E} is run on Σ_1 and determines that $c_1 = c_2$. \mathcal{S} is run on $\Sigma_2 \cup \{c_1 = c_2\}$ which produces the disjunction $b = d \vee b = e$. The procedure is now invoked recursively for $\Sigma_1 \cup \Sigma_2 \cup \{b = d\}$ and $\Sigma_1 \cup \Sigma_2 \cup \{b = e\}$. In both calls, Σ_2 produces the disjunction $g = d \vee g = e$ which is unsatisfiable. Therefore, both calls return “unsatisfiable,” hence $\Sigma \cup \mathcal{E} \cup \mathcal{S}$ is unsatisfiable.

We place one additional requirement on schemes to make the equality propagation procedure practical. Schemes must be *incremental*. This means that a scheme must be able to save its “state” when a conjunction of literals is satisfiable and it must be able to use the saved state to determine the satisfiability of larger conjunctions at incremental cost.

4 REFORMULATION

The DRAT technique as described in section 3 is severely limited by the way in which a problem is stated. Often, it is much more successful with an equivalent formulation of the problem stated in terms of a different collection of nonlogical symbols. For instance, recall the problem about family relations given in section 1. It was stated in terms of the binary relation *child*. It turns out that, given the current scheme library, the DRAT implementation is much more successful when the problem is stated in terms of *parents*, a function from an individual to his or her set of parents. One reason this formulation is better is that the library contains a scheme for a theory of fixed sized sets. DRAT discovers an instance of this scheme that allows it to remove several general clauses from the problem including one that limits the size of parent sets to two.

In an effort to circumvent this sensitivity to a problem’s formulation, DRAT is able to reformulate a problem in terms of new nonlogical symbols without changing the “meaning” of the problem. Choose-instance is often able to find scheme instances in reformulated problems where it was unable to do so in the initial formulations. DRAT’s reformulation technique is modeled after the reformulation that people do in solving analytical tasks. For an example of this refer again to the problem and diagrams given in section 1. In the diagrams appear concepts such as “married couples” and “sets of children of the same couple.” These concepts are not present in the initial problem formulation — the problem has been reformulated.

DRAT does a particular kind of reformulation called *isomorphic reformulation* in [3]. We formalize isomorphic reformulation as a relation between theories.

Definition 4 A *reformulation map* $\mathcal{R}_{\mathcal{L}_1, \mathcal{L}_2}^*$ between two languages \mathcal{L}_1 and \mathcal{L}_2 is a function from clauses

in \mathcal{L}_1 to sets of clauses in \mathcal{L}_2 .

Definition 5 A theory T_2 is an *isomorphic reformulation* of a theory T_1 just in case there exists a reformulation map $\mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*$ such that

$T_1 \models \phi \Leftrightarrow T_2 \models \mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*(\phi)$, for every clause ϕ in $\mathcal{L}(T_1)$.

If T_2 is an isomorphic reformulation of T_1 , any question we have about what clauses are entailed by T_1 can be answered by theorem proving in T_2 . Given the question, “does $T_1 \models \phi$?” we use \mathcal{R}^* to translate ϕ into $\mathcal{L}(T_2)$ and then attempt to prove that $T_2 \models \mathcal{R}^*(\phi)$.

As a simple example of isomorphic reformulation, consider the following two theories:

$$T_1 = \left\{ \begin{array}{l} R(x, x), \\ R(x, y) \Rightarrow R(y, x), \\ R(x, y) \wedge R(y, z) \Rightarrow R(x, z) \end{array} \right\}$$

$$T_2 = \left\{ \begin{array}{l} x \in R\text{-class}(x), \\ x \in R\text{-class}(y) \Rightarrow y \in R\text{-class}(x), \\ x \in R\text{-class}(y) \wedge y \in R\text{-class}(z) \Rightarrow x \in R\text{-class}(z) \end{array} \right\}$$

T_2 is an isomorphic reformulation of T_1 . To show this, we exhibit an appropriate $\mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*$. First, we introduce the function γ with $\gamma(R(x, y)) = x \in R\text{-class}(y)$ and $\gamma(\neg R(x, y)) = x \notin R\text{-class}(y)$.

The function γ is also defined in the obvious way for literals that are instances of the patterns $R(x, y)$ and $\neg R(x, y)$, i.e., given the constants a and b , $\gamma(R(a, f(b))) = a \in R\text{-class}(f(b))$.

Given the literals ϕ_1, \dots, ϕ_n , $n \geq 1$

$$\mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*(\phi_1 \vee \dots \vee \phi_n) = \{\gamma(\phi_1) \vee \dots \vee \gamma(\phi_n)\}.$$

Now $T_2 \equiv \mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*(T_1)$, using the obvious extension of \mathcal{R}^* to sets of clauses. Therefore,

$T_1 \models \phi \Leftrightarrow T_2 \models \mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*(\phi)$. To see this, note that we can take any resolution proof of $T_1 \vdash \phi$ and uniformly apply $\mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*$ to the clauses in each step of the proof to obtain a proof of $\mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*(T_1) \vdash \mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*(\phi)$. We can also define $\mathcal{R}_{\mathcal{L}(T_2), \mathcal{L}(T_1)}^*$ similarly to $\mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*$ and use it to transform any proof $\mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*(T_1) \vdash \mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*(\phi)$ into a proof of $T_1 \vdash \phi$.

5 ADDING REFORMULATION TO DRAT

One strategy for finding a satisfiability procedure for a theory T_1 is to identify a theory T_2 with the following properties: (1) a satisfiability procedure is known for T_2 , (2) we can find a reformulation map $\mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*$ demonstrating that T_2 is an isomorphic reformulation of T_1 and (3) $\mathcal{R}_{\mathcal{L}(T_1), \mathcal{L}(T_2)}^*$ is a computable function.

The actual DRAT technique is an extension of the algo-

rithm discussed in section 3 to apply the above strategy. This extension enables DRAT to generate theories that are isomorphic reformulations of T_C while searching for a set of scheme instances that is a satisfiability procedure for T_C . DRAT has a library of reformulation rules, each of which is a reformulation map. These rules are applied to an input theory T_C to construct theories that are isomorphic reformulations of T_C . The extended algorithm searches for scheme instances in these isomorphic reformulations as well as in the original T_C .

Roughly, each reformulation rule is viewed as an axiom schema that can be instantiated with nonlogical symbols and used as a rewrite rule to reformulate a theory. To understand this view, consider the following axiom schema in which R is a parameter:

$$R(x, y) \Leftrightarrow x \in F_R(y).$$

This states that for any binary relation, there is a projection function F_R that is a mapping from individuals to sets of individuals such that $F_R(y) = \{x \mid R(x, y)\}$.

DRAT can apply the above reformulation rule to binary relations in T_C . When the rule is applied to R in T_C , the new function symbol F_R is introduced and T_C is reformulated in terms of F_R . For instance, if this rule is applied to *child* in the family relations problem given earlier, it will introduce a function that we will call *parents*, from an individual to his or her set of parents. DRAT uses the formula introducing *parents*, i.e., $\text{child}(x, y) \Leftrightarrow x \in \text{parents}(y)$, to reformulate the problem, rewriting all occurrences of $\text{child}(x, y)$ to $x \in \text{parents}(y)$.

This example reformulation rule can be applied to any binary relation in any theory. More generally, DRAT's reformulation rules are conditional on properties of nonlogical symbols in a theory. A property of a nonlogical symbol is simply a first-order statement mentioning that symbol. Before giving the general form of reformulation rules, we introduce the function $\text{rf-symbols}(T)$, the set of relation and function symbols of T . The $\text{rf-symbols}(T)$ does not contain the symbols $=$ or \in , even if they are mentioned in T . These are treated as special (logical) symbols in the reformulation process.

The general form of reformulation rules is given in the following definition.

Definition 6 A triple $< P, Q, \Theta \Leftrightarrow \Psi >$ is a *reformulation rule* when it meets the following restrictions: (1) P and Q are conjunctions of clauses (both of which may be empty). (2) Θ and Ψ are conjunctions of literals. (3) $\text{rf-symbols}(P) \subseteq \text{rf-symbols}(\Theta)$ and $\text{rf-symbols}(Q) \subseteq \text{rf-symbols}(\Psi)$. (4) $\text{rf-symbols}(\Theta)$ is disjoint from $\text{rf-symbols}(\Psi)$. (5) Θ and Ψ have the same variables.

Rules are symmetric in the sense that their biconditionals can be used to introduce new symbols in "ei-

ther direction." When the parameters in Θ are instantiated with symbols in a theory T , the rule is used to reformulate T in terms of the new symbols in Ψ . The conjunction of clauses P is the condition that must be true of a theory for the reformulation rule to be used to rewrite Θ as Ψ . When the parameters in Ψ are instantiated with the symbols in T , the rule is used to reformulate T in terms of the new symbols in Θ . In this case, Q is the condition that must be true for the rule to be used.

Here is an example of a conditional reformulation rule:

$$< [x \in F(y) \Rightarrow F(y) = \{x\}], \\ [x \in F(y) \Leftrightarrow x \neq \perp \wedge x = F'(y)] >.^1$$

This rule can be applied to any theory T containing a function F whose range elements are sets of size one, i.e., $P = [x \in F(y) \Rightarrow F(y) = \{x\}]$. When applied, the rule reformulates T in terms of a function F' such that $F'(y) = x$ just in case $x \in F(y)$. Q is empty in this rule because the rule can always be applied in the other direction.

The following is an abstract description of the DRAT algorithm extended to do reformulation:

```

instances ← ∅
TI ← ∅
T'_C ← T_C
R* ← λ(t).t
UNTIL empty(T'_C) DO
  EITHER
    ref-pairs ← choose-ref-pairs(T'_C)
    IF null(ref-pairs) THEN EXIT-WITH failure
    symbols, rule ← choose(ref-pairs)
    instantiated-rule ← instantiate(rule, symbols)
    T'_C ← R(instantiated-rule, T'_C)
    R* ← λ(t).R(instantiated-rule, R*(t))
  OR
    instance ← choose-instance(T'_C)
    IF null(instance) THEN EXIT-WITH failure
    instances ← union(instance, instances)
    TI ← union(theory(instance), TI)
    FOR EACH φ ∈ T'_C
      WHEN TI ⊨ φ DO T'_C ← T'_C - φ
    END FOR
  END UNTIL

```

DRAT nondeterministically either chooses a reformulation rule and reformulates T'_C or adds the theory of the new instance to T_I . **Choose-instance** identifies an *instance* by identifying properties of the nonlogical symbols in T'_C . It looks for properties that appear in the theories of schemes. For example, when the scheme library contains a scheme one of whose axioms is $R(x, y) \Rightarrow R(y, x)$, DRAT attempts to choose instances of that scheme by looking for binary relations in T'_C that have the symmetry property.

¹The symbol \perp is used in specifying axioms about partial functions, $F(a) = \perp$ means that $F(a)$ is undefined.

Choose-ref-rule uses the identified properties of nonlogical symbols in T'_C to identify reformulation rules that can be applied to those symbols. Rules introduce new symbols as explained above. **Choose-ref-rules** returns a list of $\langle \text{symbols}, \text{rule} \rangle$ pairs, where *symbols* is an ordered list of nonlogical symbols. Each pair in the list can be applied to T_C by instantiating the parameters of the rule with *symbols*. For a rule of the form

$$\langle P, Q, \Theta \Leftrightarrow \Psi \rangle,$$

symbols can either be used to instantiate the parameters in Θ or in Ψ , but not both. Conditional rules are returned only when T'_C entails their condition. **Choose-ref-rule** guarantees that if *symbols* instantiates Θ , then P follows from T'_C ; If *symbols* instantiates Ψ , it guarantees that Q follows.

As before, if DRAT exits with T'_C empty, it has succeeded in finding a T_I equivalent to T_C ; Otherwise, it has failed.

Again we have suppressed the issues of search by giving a nondeterministic procedure. The search conducted by the extended algorithm is over a much larger space than the search conducted by the simple algorithm described in section 3. The DRAT implementation with reformulation must compare alternative problem formulations. Fortunately, we have found some effective heuristics for controlling the search. See [11] or [12] for details.

The procedure **instantiate**, instantiates a *rule* with respect to the nonlogical symbols in *symbols* to produce an *instantiated-rule*. \mathcal{R} is the reformulation procedure. We describe this procedure for the case where a rule of the form

$$\langle P, Q, \theta_1 \wedge \dots \wedge \theta_n \Leftrightarrow \Psi \rangle$$

is used to rewrite occurrences of $\theta_1 \wedge \dots \wedge \theta_n$, the *from conjunct*, to occurrences of Ψ , the *to conjunct*. The procedure for applying the rule in the other direction is obtained by reversing the biconditional and replacing references to P by references to Q .

Each set of unit clauses in T'_C of the form $\{(\theta_1)\sigma, \dots, (\theta_n)\sigma\}$, where σ is a substitution for the variables in the θ_i , is rewritten as the set of unit clauses $(\Psi)\sigma$. Each clause containing the literals $(\neg\theta_1)\sigma, \dots, (\neg\theta_n)\sigma$ is rewritten to contain $(\neg\Psi)\sigma$. After all possible occurrences are rewritten, the clauses in Q are added to the rewritten theory.

We call a rewriting produced by \mathcal{R} *complete* when it removes all of the nonlogical symbols appearing in the from conjunct. \mathcal{R} may or may not produce a complete rewriting. For example, given a right hand side of the form $R(f(x))$, rewriting will only be complete when R and f appear in a theory only in patterns of this form. If the rewriting process is not complete, \mathcal{R} adds the instantiated $\Theta \Leftrightarrow \Psi$ to the rewritten theory.

As an example of applying \mathcal{R} , consider again the rule

$$\begin{aligned} & \langle [x \in F(y) \Rightarrow F(y) = \{x\}], \\ & \quad [x \in F(y) \Leftrightarrow x \neq \perp \wedge x = F'(y)] \rangle. \end{aligned}$$

As noted, the condition P must follow from a theory to reformulate F as F' in that theory. Since the condition Q is empty, there are no clauses to add to the resulting theory. If the rewriting is not complete, $[x \neq \perp \wedge x = F'(y) \Leftrightarrow x \in F(y)]$ is added to the rewritten theory. Since there is no condition Q , this rule can always be used, in the other direction, to reformulate F' as F . In this case, P is added to the rewritten theory. Again, the biconditional may need to be added to the rewritten theory.

To ensure that the extended DRAT algorithm generates only isomorphic reformulations, each reformulation rule must be shown to generate only isomorphic reformulations. To guarantee this, we require that, when instantiated, each reformulation rule be an *extending definition*.

Definition 7 A reformulation rule $\langle P, Q, \Theta \Leftrightarrow \Psi \rangle$ is an *extending definition* if for all theories T the following conditions hold:

1. Whenever the $\text{rf-symbols}(\Theta) \subseteq \text{rf-symbols}(T)$, $\text{rf-symbols}(\Psi)$ is disjoint from $\text{rf-symbols}(T)$ and $T \models P$, then every model of T can be expanded to a model of $T \cup \{\Theta \Leftrightarrow \Psi\}$.
2. Whenever the $\text{rf-symbols}(\Psi) \subseteq \text{rf-symbols}(T)$, $\text{rf-symbols}(\Theta)$ is disjoint from $\text{rf-symbols}(T)$ and $T \models Q$, then every model of T can be extended to a model of $T \cup \{\Theta \Leftrightarrow \Psi\}$.

Section 6 shows that for any reformulation rule *rule*, $\lambda(t).\mathcal{R}(\text{rule}, t)$ is a computable function and so long as *rule* is an extending definition, that whenever a theory T entails the appropriate condition of *rule*, $\mathcal{R}(\text{rule}, T)$ is an isomorphic reformulation of T .

The \mathcal{R}^* produced by DRAT on the problem $\langle \Sigma, T_C, \Phi \rangle$ is the composition of reformulation maps used by the algorithm to reformulate T_C . Since each reformulation map generates an isomorphic reformulation, $\mathcal{R}^*(T_C)$ is an isomorphic reformulation of T_C . Since each step is computable, \mathcal{R}^* is a computable function.

Finally we point out that, since Ψ and Θ in the reformulation rule $\langle P, Q, \Theta \Leftrightarrow \Psi \rangle$ are required to have the same variables, $\mathcal{R}^*(\Sigma)$ and $\mathcal{R}^*(\Phi)$ will always be ground. However, even though Σ and Φ are conjunctions of ground literals, $\mathcal{R}^*(\Sigma)$ and $\mathcal{R}^*(\Phi)$ may not be. To see this, suppose that Σ contains the literal $\neg\phi$ and $\mathcal{R}^*(\phi)$ is a conjunction. Then $\neg\mathcal{R}^*(\phi)$ will be a disjunction.

Section 6 shows that when DRAT uses reformulation in designing a satisfiability procedure for a problem $\langle \Sigma, T_C, \Phi \rangle$ and $\mathcal{R}^*(\Sigma)$ is a conjunction of literals, the problem can be solved by solving

$\langle \mathcal{R}^*(\Sigma), \mathcal{R}^*(T_C), \mathcal{R}^*(\Phi) \rangle$. The fact that a satisfiability procedure for a reformulation of a problem requires $\mathcal{R}^*(\Sigma)$ to be a conjunction of literals is not a significant difficulty in the more general setting discussed in section 7 in which satisfiability procedures are used in conjunction with a theorem prover.

5.1 AN EXAMPLE

In practice, we have found that adding reformulation to DRAT increases its effectiveness considerably. We illustrate this with a relatively simple example excerpted from the DRAT implementation design of a satisfiability procedure for the example problem given in section 1. We illustrate the implementation's behavior on the set T of clauses:

$$\begin{aligned} &\neg\text{married}(x, x), \\ &\text{married}(x, y) \Rightarrow \text{married}(y, x) \\ &\text{married}(x, y) \wedge \text{married}(y, z) \Rightarrow \neg\text{married}(x, z) \\ &\text{married}(y, x) \wedge \text{married}(z, x) \Rightarrow y = z \end{aligned}$$

There are three schemes in DRAT's library that are relevant to the example. The scheme $\hat{\mathcal{F}}$ for the theory of partial 1-1 functions with parameters F and F' , which are inverse functions, and $\text{theory}(\hat{\mathcal{F}}) = \{x = F(y) \wedge x \neq \perp \Leftrightarrow y = F'(x) \wedge y \neq \perp\}$; The scheme \mathcal{S}_2 for the theory of sets of size two with S as a parameter and $\text{theory}(\mathcal{S}_2) = \{x_1 \in S \wedge x_2 \in S \wedge x_1 \neq x_2 \Rightarrow S = \{x_1, x_2\}\}$; And, the scheme \mathcal{E} for the theory of equality with uninterpreted function symbols.

The relevant reformulation rules are:

$$\begin{aligned} r_1 &= \langle \cdot, \cdot, R(x, y) \Leftrightarrow y \in F_R(x) \rangle \\ r_2 &= \langle x \in F(y) \Rightarrow F(y) = \{x\}, \\ &\quad [x \in F(y) \Leftrightarrow x \neq \perp \wedge x = F'(y)] \rangle \\ r_3 &= \langle (x \neq \perp \wedge y \neq \perp) \Rightarrow x = F(y) \Leftrightarrow y = F(x), \\ &\quad [x = F(y) \wedge x \neq \perp \Leftrightarrow F'(y) = \{x, y\} \wedge x \neq y] \rangle \end{aligned}$$

As is typical in the implementation, these rules are normally used only in one direction. As noted in section 5, r_1 reformulates a binary relation in a theory as a function F_R onto sets: $F_R(x) = \{y \mid R(x, y)\}$. Also as noted in section 5, when applied to a theory containing a function F whose range elements are sets of size one, r_2 introduces a function F' such that $F'(y) = x$ just in case $x \in F(y)$. The rule r_3 reformulates an F that is its own inverse as a function F' , mapping an individual into sets of size two such that $F'(x) = \{x, F(x)\}$.

Given the schemes above, DRAT is unable to design a satisfiability procedure for T without reformulation. In an effort to design a satisfiability procedure for all of T , the DRAT implementation repeatedly reformulates the problem, finally producing a formulation in terms of a function that we will call *couple*, mapping an individual to the married couple of which he or she is a member.

DRAT uses rule r_1 to reformulate T in terms of a function that we will call *spouses*, a mapping from an individual to the set of his or her spouses. $\mathcal{R}(r_1, T)$ is

$$\begin{aligned} &x \notin \text{spouses}(x), \\ &x \in \text{spouses}(y) \Rightarrow y \in \text{spouses}(x) \\ &x \in \text{spouses}(y) \wedge y \in \text{spouses}(z) \Rightarrow x \notin \text{spouses}(z) \\ &y \in \text{spouses}(x) \wedge z \in \text{spouses}(x) \Rightarrow y = z \end{aligned}$$

DRAT uses rule r_2 to reformulate $\mathcal{R}(r_1, T)$ in terms of a partial function that we will call *spouse*, a mapping from an individual to his or her spouse. $\mathcal{R}(r_2, \mathcal{R}(r_1, T))$ is

$$\begin{aligned} &x \neq \text{spouse}(x) \vee x = \perp, \\ &x = \text{spouse}(y) \wedge x \neq \perp \Rightarrow y = \text{spouse}(x) \wedge y \neq \perp \\ &x = \text{spouse}(y) \wedge x \neq \perp \wedge y = \text{spouse}(z) \wedge y \neq \perp \\ &\quad \Rightarrow x \neq \text{spouse}(z) \vee x = \perp \\ &y = \text{spouse}(x) \wedge y \neq \perp \wedge z = \text{spouse}(x) \wedge z \neq \perp \\ &\quad \Rightarrow y = z \end{aligned}$$

Note that the second and fourth clauses in this set follow from instances of $\hat{\mathcal{F}}$ and \mathcal{E} respectively. Hence, if DRAT were to terminate at this point, T'_C would include only the first and third clauses.

DRAT uses rule r_3 to reformulate the above theory in terms of the function *couple*. The result is

$$\begin{aligned} &\text{couple}(x) \neq \{x, x\} \vee x = x, \\ &\text{couple}(x) = \{x, y\} \wedge x \neq y \Rightarrow \\ &\quad \text{couple}(y) = \{y, x\} \wedge y \neq x, \\ &\text{couple}(x) = \{x, y\} \wedge x \neq y \wedge \\ &\quad \text{couple}(y) = \{y, z\} \wedge y \neq z \Rightarrow \\ &\quad \text{couple}(x) \neq \{x, z\} \vee x = z, \\ &\text{couple}(y) = \{x, y\} \wedge y \neq x \wedge \\ &\quad \text{couple}(z) = \{z, x\} \wedge z \neq x \Rightarrow y = z \end{aligned}$$

All of the clauses in this set follow from the combination of \mathcal{S}_2 and an instance of \mathcal{E} containing the uninterpreted function symbol *couple*. Thus, through the use of reformulation, DRAT succeeds in designing a satisfiability procedure for the theory T . Without reformulation it is unable to design a procedure for any subset of T .

6 STEPS TOWARDS THE COMPLETENESS OF DRAT

This section proves two results towards the completeness of DRAT. First, we show that DRAT designs satisfiability procedures. If DRAT successfully designs a procedure for some set of axioms T_C , then that procedure can be used to decide the problem $\langle \Sigma, T_C, \Phi \rangle$ for any conjunctions of ground literals Σ and Φ . Second, we consider the addition of reformulation to DRAT and show that a satisfiability procedure for $\mathcal{R}^*(T_C)$ can be used as a satisfiability procedure for T_C so long as $\mathcal{R}^*(\Sigma)$ is a conjunction of literals. These results are necessary preliminaries for the proof of completeness in section 7.

6.1 DRAT DESIGNS SATISFIABILITY PROCEDURES

Before proceeding to prove that DRAT designs satisfiability procedures, we recall properties of schemes pre-

sented thus far and discuss some additional required properties.

Recall that a scheme for a theory T is a procedure that decides the satisfiability of $\Sigma \cup T$, where Σ is a conjunction of ground literals. Given a particular Σ , each scheme also computes the set of equalities between constants in Σ that follow from $\Sigma \cup T$. If T is nonconvex, its scheme also computes disjunctions of equalities between constants in Σ that follow from $\Sigma \cup T$.

We call a first-order theory whose formulas contain no existential quantifiers a *quantifier-free* theory. An additional requirement on schemes is that their theories be quantifier-free. As a practical matter, this is not a serious restriction beyond restricting schemes to be tractable. See [8] for further discussion of this point.

The theories of schemes are also required to have infinite models. The equality propagation technique may not work if a theory has only finite models because, given a set of constant symbols larger than the set of individuals in the model's domain, such a theory implies the disjunction of equalities between those constant symbols. Theories with infinite models do not imply disjunctions of equalities between variables. Therefore, given a theory T with infinite models, such disjunctions can only follow from $T \cup \Sigma$, for some Σ whose satisfiability is being decided. Any disjunctions of equalities between constants that follow must involve only constants mentioned in Σ . This restriction to theories with infinite models does not appear to be significant. To date, we have not found any schemes that we could not include because they violated this restriction.

The theorem proved below is similar to the theorem given in [5]. It differs in the addition of the requirement that each scheme's theory have infinite models. The theorem appearing in [5] is incorrectly stated. The reason a different proof is included here is that the proof given in [5] is incorrect.² We also include our proof because the technique is much more direct and serves as a foundation for research in progress to extend our results.

Theorem 1 *Let T_1 and T_2 be theories with no common nonlogical symbols. If there are schemes for T_1 and T_2 , there is a scheme for $T_1 \cup T_2$.*

Proof: We prove that the procedure described in section 3 for combining two schemes is a scheme for $T_1 \cup T_2$. If the scheme for T_1 or T_2 reports "unsatisfiable," clearly $\Sigma_1 \cup \Sigma_2 \cup T_1 \cup T_2$ is unsatisfiable and, since $\Sigma_1 \cup \Sigma_2$ and Σ are cosatisfiable, $\Sigma \cup T_1 \cup T_2$ is unsatisfiable. We must show that if the procedure of section 3 reports "satisfiable," $\Sigma \cup T_1 \cup T_2$ is satisfiable.

²A correct version of the theorem appears in [7], however, the proof given there is still incorrect.

This is done by showing how to construct a model of $\Sigma \cup T_1 \cup T_2$ when the procedure reports "satisfiable."

Let $C = \{c_0, \dots, c_n\}$ be the set of constant symbols appearing in Σ_1 or Σ_2 . Let E be the set of equalities propagated by the procedure of section 3. As we will see, when the procedure halts, E contains all the $c_1 = c_2$ such that $c_1, c_2 \in C \wedge \Sigma_1 \cup \Sigma_2 \cup T_1 \cup T_2 \models c_1 = c_2$. E will also contain any equalities chosen when case splitting occurs.

Let $\bar{E} = \{c_1 = c_2 \mid c_1, c_2 \in C \wedge c_1 = c_2 \notin E\}$. Since the schemes for T_1 and T_2 reported "satisfiable," there are models of $\Sigma_1 \cup T_1 \cup E$ and $\Sigma_2 \cup T_2 \cup E$. Let \mathcal{M}_1 and \mathcal{M}_2 be models of $\Sigma_1 \cup T_1 \cup E$ and $\Sigma_2 \cup T_2 \cup E$ respectively that agree on the interpretation of the equalities in \bar{E} . We show how to construct a model $\mathcal{M} \models \Sigma \cup T_1 \cup T_2$ from \mathcal{M}_1 and \mathcal{M}_2 .

Before giving this construction, we show that it is possible to pick an \mathcal{M}_1 and \mathcal{M}_2 that agree on \bar{E} . First note that if \bar{E} is empty, all \mathcal{M}_1 and \mathcal{M}_2 agree. Now suppose that \bar{E} is not empty. In this case, there exists an \mathcal{M}_1 and an \mathcal{M}_2 that do not satisfy any equality in \bar{E} . For suppose to the contrary. In particular, suppose that every \mathcal{M}_1 satisfies some equality in \bar{E} . If \bar{E} contains exactly one equality, $c_1 = c_2$, $\Sigma_1 \cup T_1 \cup E \models c_1 = c_2$ and $c_1 = c_2 \in E$, not \bar{E} . If \bar{E} contains more than one equality, $\Sigma_1 \cup T_1 \cup E$ entails the disjunction of equalities in \bar{E} . But then $\Sigma_1 \cup T_1 \cup E$ is nonconvex which is impossible because, instead of returning satisfiable, the algorithm in section 3 would have case split in this situation. This same argument can be made for \mathcal{M}_2 and, hence, there exists an \mathcal{M}_2 that does not satisfy any of the equalities in \bar{E} . Thus, we can choose an \mathcal{M}_1 and \mathcal{M}_2 that agree on the interpretation of the equalities in \bar{E} .

Note that since \mathcal{M}_1 and \mathcal{M}_2 agree on the interpretation of the equalities in E and in \bar{E} , they agree on the interpretation of every equality between constants in C .

Let $\mathcal{M}_1 = \langle D_1, R_1, F_1, C_1 \rangle$, where D_1 is the domain of \mathcal{M}_1 , R_1 is the interpretation of relation symbols of \mathcal{M}_1 in D_1 , F_1 is the interpretation of the functions symbols of \mathcal{M}_1 and C_1 is the interpretation of individual constant symbols in \mathcal{M}_1 . Similarly, let $\mathcal{M}_2 = \langle D_2, R_2, F_2, C_2 \rangle$.

We now construct \mathcal{M} by merging \mathcal{M}_1 and \mathcal{M}_2 as follows. The domain of \mathcal{M} is $D_1 \cup D'_2$, where D'_2 is the domain of \mathcal{M}'_2 , a modified version of \mathcal{M}_2 . \mathcal{M}'_2 is obtained by replacing individuals in D_2 by individuals in D_1 when they are designated by the same constant symbol. For all constant symbols $c \in C$, replace every occurrence of $C_2(c)$ in D_2 by $C_1(c)$, i.e., $C'_2(c) = C_1(c)$ when c is a shared constant symbol and $C'_2(c) = C_2(c)$ otherwise. For all R in the domain of R_2 , let $R'_2(R)$ be the set $R_2(R)$ modified by the above replacement procedure. Similarly, let F'_2 be

the new interpretation of the function symbols of \mathcal{M}_2 . $\mathcal{M}_2' = \langle D'_2, R'_2, F'_2, C'_2 \rangle$.

\mathcal{M}_2 and \mathcal{M}_2' are isomorphic structures because \mathcal{M}_1 and \mathcal{M}_2 agree on the interpretation of every equality between constants in C . If \mathcal{M}_1 and \mathcal{M}_2 did not agree, then \mathcal{M}_2 and \mathcal{M}_2' would not be isomorphic. For suppose, that $\mathcal{M}_1 \models c_1 = c_2$ but $\mathcal{M}_2 \not\models c_1 = c_2$. Then the two constant symbols designate the same individual in D'_2 and different individuals in D_2 and, hence, \mathcal{M}_2' is not isomorphic to \mathcal{M}_2 .

To finish the construction of \mathcal{M} , we take $\mathcal{M} = \langle D_1 \cup D'_2, R_1 \cup R'_2, F_1 \cup F'_2, C_1 \cup C'_2 \rangle$. Since $\mathcal{M}_1 \models \Sigma_1 \cup T_1$ and $\mathcal{M}_2 \models \Sigma_2 \cup T_2$, $\mathcal{M} \models \Sigma_1 \cup \Sigma_2 \cup T_1 \cup T_2$. Since $\Sigma_1 \cup \Sigma_2$ and Σ are cosatisfiable, $\mathcal{M} \models \Sigma \cup T_1 \cup T_2$ and the proof of the theorem is complete. \square .

The fact that DRAT designs satisfiability procedures is a direct consequence of theorem 1. Since the result of combining two schemes is again a scheme, any number of schemes can be combined by this method.

6.2 DRAT DOES ISOMORPHIC REFORMULATION

This section includes the proofs of two properties of DRAT's reformulation procedure \mathcal{R} . These results are sufficient to show how a satisfiability procedure generated by DRAT for some reformulated theory can be used to solve the original problem.

Lemma 1 *If a reformulation rule (rule) is an extending definition in T of the form $\langle P, Q, \Theta \Leftrightarrow \Psi \rangle$ and $T \models P$, then $\mathcal{R}(\text{rule}, T)$ is an isomorphic reformulation of T .*

Proof: The condition that must be met is that if $T \models P$, $T \models \phi \Leftrightarrow \mathcal{R}(\text{rule}, T) \models \mathcal{R}(\text{rule}, \phi)$, for any clause $\phi \in \mathcal{L}(T)$. We prove the equivalent fact that if $T \models P$, $SAT(T \cup \{\neg\phi\}) \Leftrightarrow SAT(\mathcal{R}(\text{rule}, T) \cup \neg\mathcal{R}(\text{rule}, \phi))$, where $SAT(T)$ means that T is satisfiable.

[\Rightarrow] If $SAT(T \cup \{\neg\phi\})$, $SAT(T \cup \{\Theta \Leftrightarrow \Psi\} \cup \{\neg\phi\})$ because, by the definition of extending definition, every model of T can be extended to a model of $T \cup \{\Theta \Leftrightarrow \Psi\}$. Therefore, there exists a model of $T \cup \{\Theta \Leftrightarrow \Psi\} \cup \{\neg\phi\}$. But

$T \cup \{\Theta \Leftrightarrow \Psi\} \cup \{\neg\phi\} \models \mathcal{R}(\text{rule}, T) \cup \neg\mathcal{R}(\text{rule}, \phi)$.

Hence every model of $T \cup \{\Theta \Leftrightarrow \Psi\} \cup \{\neg\phi\}$ is a model of $\mathcal{R}(\text{rule}, T) \cup \neg\mathcal{R}(\text{rule}, \phi)$. Since there exists a model of $T \cup \{\Theta \Leftrightarrow \Psi\} \cup \{\neg\phi\}$, there exists a model of $\mathcal{R}(\text{rule}, T) \cup \neg\mathcal{R}(\text{rule}, \phi)$ and hence, it is satisfiable.

[\Leftarrow] The proof in this direction is similar, with the added step of showing that every model of $\mathcal{R}(\text{rule}, T) \cup \neg\mathcal{R}(\text{rule}, \phi)$ can be extended to a model of $\mathcal{R}(\text{rule}, T) \cup \{\Theta \Leftrightarrow \Psi\} \cup \neg\mathcal{R}(\text{rule}, \phi)$. Since rule is an extending definition, every model of a theory T_1 that entails Q can be extended to a model of $T_1 \cup \{\Theta \Leftrightarrow \Psi\}$.

By the definition of \mathcal{R} , the clauses of Q will appear in $\mathcal{R}(\text{rule}, T)$ and hence $\mathcal{R}(\text{rule}, T) \models Q$. Therefore, every model of $\mathcal{R}(\text{rule}, T)$ can be extended to a model of $\mathcal{R}(\text{rule}, T) \cup \{\Theta \Leftrightarrow \Psi\}$. Thus, if $\mathcal{R}(\text{rule}, T) \cup \neg\mathcal{R}(\phi)$ is satisfiable, so is $T \cup \{\neg\phi\}$. \square

It follows directly from this lemma and the fact that extending definitions can be used in either direction, that a reformulation rule $(P \wedge Q) \Rightarrow [\Theta \Leftrightarrow \Psi]$ with the $rf\text{-symbols}(\Psi)$ instantiated in term of a theory T can be used to reformulate T in terms of Θ so long as $T \models Q$.

Lemma 2 *For any reformulation rule (rule), the function $\lambda(t).\mathcal{R}(\text{rule}, t)$ is computable.*

Proof: Suppose the biconditional of rule is $\Theta \Leftrightarrow \Psi$ and \mathcal{R} applies rule to rewrite occurrences of Ψ to occurrences of Θ in T , as described in section 5. Since $rf\text{-symbols}(\Theta)$ are disjoint from $rf\text{-symbols}(T)$, a rewrite step can never introduce a pattern of literals to which rule can be applied a second time. The rewrite is applied repeatedly until one of the following events occurs: (1) all of the symbols in $rf\text{-symbols}(\Psi)$ are removed from T or (2) no new occurrences of Ψ can be found, even though symbols in $rf\text{-symbols}(\Psi)$ are still present. In either case, repeated application of the rewrite rule terminates. Hence, $\lambda(t).\mathcal{R}(\text{rule}, t)$ is computable. \square

The two preceding lemmas are sufficient to show that a satisfiability procedure for $\mathcal{R}^*(T_C)$ can be used to solve the problem $\langle \Sigma, T_C, \Phi \rangle$, so long as $\mathcal{R}^*(\Sigma)$ is a conjunction of ground literals. Assuming that $\mathcal{R}^*(\Sigma)$ is a conjunction, the satisfiability procedure is used to solve the problem by solving $\langle \mathcal{R}^*(\Sigma), \mathcal{R}^*(T_C), \mathcal{R}^*(\Phi) \rangle$ as follows. For each $\phi \in \Phi$, if $\neg\mathcal{R}^*(\phi)$ is a conjunction of literals, we use the procedure to determine if $\mathcal{R}^*(\Sigma) \cup \mathcal{R}^*(T_C) \cup \neg\mathcal{R}^*(\phi)$ is unsatisfiable. This is the case if and only if $\Sigma \cup T_C \cup \neg\phi$ is unsatisfiable. If $\neg\mathcal{R}^*(\phi)$ is a disjunction of literals, the procedure is used to determine the satisfiability of $\mathcal{R}^*(\Sigma) \cup \mathcal{R}^*(T_C) \cup l$, for each literal $l \in \neg\mathcal{R}^*(\phi)$. If any of these is satisfiable, $\mathcal{R}^*(\Sigma) \cup \mathcal{R}^*(T_C) \cup \neg\mathcal{R}^*(\phi)$ is satisfiable; otherwise it is unsatisfiable.

7 THE COMPLETENESS OF DRAT

Two simplifying assumptions were made in the previous sections. First, in definition 3, it was assumed that a problem for DRAT was of a restricted form. Second, it was assumed that DRAT's success depended on designing a satisfiability procedure for all of T_C . Both of these assumptions are now relaxed and we show how a literal satisfiability procedure is interfaced with a resolution theorem prover in such a way that the procedure/theorem prover combination is complete.

A problem for DRAT is now taken to be a pair $\langle \Gamma, \phi \rangle$, where Γ is a set of first-order formulas and ϕ is a first-

order formula. A pair $\langle \Gamma, \phi \rangle$ is interpreted as the question, " $\Gamma \models \phi?$ "

As a typical preprocessing step for resolution theorem proving, Γ and $\neg\phi$ are converted to sets of clauses which will be called Γ' and $\neg\phi'$ respectively. Let T_C be the set of nonground clauses in Γ' . As before, DRAT is used to design a literal satisfiability procedure for T_C . However, instead of exiting with failure if it is unable to design a procedure for all of T_C , it returns the satisfiability procedure and T'_C , those clauses not incorporated into the satisfiability procedure. Also, as before, DRAT returns the reformulation map \mathcal{R}^* .

The algorithm given in section 5 refers to the set of clauses for which a literal satisfiability procedure has been designed as T_I . Here that procedure is referred to as S_{T_I} . We show how S_{T_I} is used along with a resolution theorem prover to demonstrate the unsatisfiability of $Cl = \mathcal{R}^*(\Gamma') \cup \mathcal{R}^*(\neg\phi')$. The nonground clauses of Cl are manipulated by the theorem prover in the usual way, except that clauses in T_I are prohibited from resolving with ground clauses. These resolutions are unnecessary because S_{T_I} is a "compression" of any resolution steps that can result from such a resolvent.

S_{T_I} is used in the manipulation of ground clauses in Cl and ground clauses derived from Cl during theorem proving. It is interfaced to the theorem prover via *theory resolution*[9]. One type of theory resolution, called *total narrow* theory resolution, requires a decision procedure for a theory T , given a set of literals L , to compute subsets L' of L such that $L' \cup T$ is unsatisfiable. Such a procedure is used to compute T -resolvents of a set of clauses as follows. Consider the decomposition of the clauses into $K_i \vee L_i$, where each K_i is a single literal in $\mathcal{L}(T)$ and L_i is disjunction of literals (possibly empty). For each subset of the K_i , say $\{K_{i_1}, \dots, K_{i_n}\}$, that is unsatisfiable in T , the clause $L_1 \vee \dots \vee L_n$ is a T -resolvent.

The theorem prover constructs T_I -resolvents from ground clauses, using S_{T_I} to compute sets of ground literals that are unsatisfiable in T_I . Let GrL be the set of ground unit clauses in Cl and let $GrCl$ be the set of ground nonunit clauses in Cl . First, the ground clauses are separated into clauses that are in $\mathcal{L}(T_I)$ and clauses that are not. This is accomplished for the clauses in GrL using the procedure described in section 3; It is accomplished for clauses in $GrCl$ in a similar fashion.

If a ground clause c_1 contains a literal that is not in $\mathcal{L}(T_I)$ and a ground clause c_2 contains the negation of that literal, the theorem prover computes the resolvent of c_1 and c_2 in the normal way. T_I -resolvents are computed using S_{T_I} to compute sets of ground literals that are unsatisfiable in T_I as follows. Let GrL_{T_I} be the set of literals in GrL that are in $\mathcal{L}(T_I)$. Let $GrCl_{T_I}$ be the set of literals in $\mathcal{L}(T_I)$ appearing in clauses of $GrCl$. We input progressively larger sub-

sets of $GrLits = GrL_{T_I} \cup GrCl_{T_I}$ to S_{T_I} , as long as those sets are satisfiable in T_I . Once a set is unsatisfiable in T_I , all supersets of it will also be unsatisfiable. When the theorem prover deduces a new ground literal in GrL_{T_I} , it is added to $GrLits$. The smallest subsets of $GrLits$ found to be unsatisfiable in T_I are used to compute T_I -resolvents of ground clauses.

Theorem 2 *Given the problem $\langle \Gamma, \phi \rangle$, let S_{T_I} be a literal satisfiability procedure for $T_I \subseteq \mathcal{R}^*(\Gamma)$. If $\Gamma \models \phi$, S_{T_I} combined with the theorem prover will demonstrate the unsatisfiability of Cl .*

Proof: In [9], Stickel shows that, given a set of clauses $K_i \vee L_i$, if a decision procedure for a theory T computes all subsets K_i that are *minimally* unsatisfiable in T , total narrow theory resolution is complete. We must show that the above procedure for computing T_I -resolvents computes all subsets of $GrLits$ that are minimally unsatisfiable in T_I . Clearly, so long as S_{T_I} is a literal satisfiability procedure, the above procedure computes all these subsets. Thus, the completeness result follows directly from the results of section 6. \square

The procedure described above can be made much more efficient. There are several refinements used by the DRAT implementation to consider far fewer subsets for unsatisfiability in T_I . We discuss two of these here. One refinement is to distinguish between literals in GrL_{T_I} and $GrCl_{T_I}$. First, we consider the satisfiability of GrL_{T_I} . If this is unsatisfiable, we are done. Otherwise, we consider progressively larger sets of literals appearing in clauses in $GrCl_{T_I}$. For each such set s , S_{T_I} is used to determine whether or not $GrL_{T_I} \cup s$ is unsatisfiable in T_I .

Note that the subsets identified with this refinement are not always minimal: it is possible for a subset of $GrCl_{T_I}$ union a subset of GrL_{T_I} to be unsatisfiable in T_I . However, it turns out that completeness of theory resolution is retained in this case, since the extraneous literals are in GrL_{T_I} and, therefore, are unit clauses.

A second simpler refinement only considers subsets of $GrCl_{T_I}$ each of whose elements appears in a different clause in $GrCl$.

As a final point about the efficiency of the procedure for computing subsets that are minimally unsatisfiable in T_I , recall that schemes are required to be incremental. Because of this, S_{T_I} is used very efficiently to consider progressively larger sets of literals.

It is often most effective to leverage the use of S_{T_I} by doing as much of the theorem proving as possible at the "ground level." The DRAT implementation uses "set of support" strategy which is very effective in accomplishing this when $\neg\phi'$ is ground because it tends to produce ground resolvents.

8 Summary and Ongoing Work

We have presented a formalization of DRAT: a technique for automatic design of satisfiability procedures. We have shown how these procedures are interfaced to a theorem prover so that it can, in many cases, prove theorems more efficiently. Given Ψ , the set of axioms of a problem, and $S_{\Psi'}$, a literal satisfiability procedure designed for $\Psi' \subseteq \Psi$, we have proven that for any first-order statement ϕ , if $\Psi \models \phi$, the theorem prover/ $S_{\Psi'}$ combination will prove ϕ .

The major steps of our argument were as follows:

1. We showed that a combination of satisfiability procedures with certain properties is again a satisfiability procedure.
2. We showed that the reformulation that is essential to DRAT's effectiveness is isomorphic reformulation and, therefore, a satisfiability procedure of a reformulated theory can be used to solve problems in the original theory.
3. We proved the completeness of our technique for combining literal satisfiability procedures with a theorem prover. In this combination, $S_{\Psi'}$ is used to compute Ψ' -resolvents from ground clauses and the theorem prover is restricted so that it does not resolve ground clauses on literals in $\mathcal{L}(\Psi')$.

In our ongoing work, we are attempting to extend DRAT's scheme combination technique. As much as possible, we would like to remove the restriction on the sharing of nonlogical symbols between component scheme instances in combinations. We are exploring the conditions under which limited types of overlap between nonlogical symbols is allowed. When overlap is allowed, component schemes must propagate more information than just equalities between constant symbols. In most cases where overlap is allowed and in which the schemes propagate at least the set of equalities between constants, it is not difficult to show the completeness of a propagation technique. The major issue that arises is proving that the propagation terminates.

As an example, consider allowing two schemes to share function symbols. The schemes must propagate all equalities between ground terms involving shared function symbols. The proof technique used in section 6.1 can be extended to prove that such schemes combined by an appropriately extended propagation technique will produce semi-decision procedures for the combinations of their theories. However, in general, it is not possible to prove that the propagation will terminate.

One situation in which overlap is allowed occurs when the theories of schemes are sets of clauses in a sorted first-order logic. In this case, a function symbol F whose range is disjoint from its domain can be shared between schemes because terms of the form $F(F(x))$

are not well formed and, hence, it is easy to show that propagation of terms involving F will terminate.

Acknowledgements

Richard Fikes, Bob Nado, Mike Lowry, and David McAllester provided helpful comments on drafts. Dave McAllester pointed out the error in Nelson & Oppen's proof of the combination of satisfiability procedures and suggested the technique we used in our proof. Bob Nado participated in numerous discussions on many aspects of the paper.

References

- [1] Brachman, R.J., Fikes, R.E. and Levesque, H.J., "KRYPTON: A Functional Approach to Knowledge Representation," in Brachman, R.J. and Levesque, H.J. (editors), Readings in Knowledge Representation, pp. 411-429, Morgan Kaufmann, 1985.
- [2] Cohn, A.G., "Many Many Sorted Logics," Workshop on Principles of Hybrid Reasoning, pp. 63-78, 1988.
- [3] Korf, R.E., "Toward a Model of Representation Changes," *Artificial Intelligence*, 14, pp. 41-78, 1980.
- [4] Loveland, D.W., Automated Theorem Proving: a logical basis, North Holland, 1978.
- [5] Nelson, G. and Oppen, D.C., "Simplification by Cooperating Decision Procedures," *ACM Transactions on Programming Languages and Systems*, 1, pp. 245-257, 1979.
- [6] Nelson, G. and Oppen, D.C., "Fast Decision Procedures Based on Congruence Closure," *Journal of the ACM*, 27, pp. 356-364, 1980.
- [7] Nelson, G., "Combining Satisfiability Procedures by Equality-Sharing," in Bledsoe, W.W. and Loveland, D.W., Automated Theorem Proving: After 25 Years, American Mathematical Society, 1984.
- [8] Oppen, D.C., "Complexity, Convexity and Combinations of Theories," *Theoretical Computer Science*, 12, pp. 291-302, 1980.
- [9] Stickel, M.E., "Automated Deduction by Theory Resolution," *Automated Reasoning*, 1, pp. 333-355, Reidel Publishing Co., 1985.
- [10] Van Baalen, J. and Davis, R., "Overview of an Approach to Representation Design," AAAI88, pp. 392-397, 1988.
- [11] Van Baalen, J., "Toward a Theory of Representation Design," MIT Artificial Intelligence Laboratory, Technical Report 1128, 1989.
- [12] Van Baalen, J., "Automated Design of Specialized Representations," to appear in *Artificial Intelligence*.
- [13] Wos, L., Overbeek, R., Lusk, E., and Boyle, J., Automated Reasoning, Prentice-Hall, 1984.

The Feasibility of Defeat in Defeasible Reasoning

Gerard Vreeswijk

Vrije Universiteit Amsterdam,
Faculty of Mathematics and Informatics,
De Boelelaan 1081a, NL-1081 HV Amsterdam.

Email address: vreesw@cs.vu.nl

Abstract

Systems of defeasible reasoning are characterized by defeasible proofs, called *arguments*. I claim that sensible criteria of defeat among arguments in those systems are feasible, be it to a certain extent. As defeat eventually becomes unenforceable, the only option left is to pursue both arguments concurrently. This paper tries to confine the reach of defeat among arguments by means of several case studies, some of which are taken from the literature.

1. INTRODUCTION

Reasoning beyond the information enclosed in the premises is a tempting but risky activity. It is tempting, because sheer deductive reasoning brings us no more than what was already recorded in the premises. And it is risky, because we might jump to the wrong conclusions. This is, very briefly, the issue of ampliative inference mechanisms. Ampliative inference can be defined as the result of rational non-deterministic non-monotonic reasoning (Loui, 1990). The term itself is suggested by the American philosopher Peirce (1839-1914).

The subject of ampliative inference has cropped up several times in the history of logic, recently under the names *nonmonotonic reasoning* (Ginsberg, 1987) and *defeasible reasoning* (Pappas and Swain, 1978) and

(Pollock, 1987). The discipline of nonmonotonic reasoning is, as I see it, inspired by the closed-world assumption: everything what is not derivable is not true. On the other hand, the school of defeasible reasoning might be considered as the formal philosophical counterpart of the theory of argumentation. Some recent contributions using the best of both worlds are (Loui, 1987), (Konolige, 1988), (Nute, 1988), and (Lin and Shoham, 1989).

A formal system of defeasible argumentation enables proofs of varying conclusive force. The idea is that, in case of conflicting proofs, the best proof remains in force to deliver the authoritative conclusion. As Konolige (1988) remarks, "the analysis of defeat is the most subtle and challenging part of an argumentation system." Here, notions like *most specific explanation*, *shortest proof*, and *strongest evidence* typically come into play.

It is important to note that many systems of defeasible argumentation operate on a single scenario. That is, reasoning is supposed to lead us from one world to another. Such a single scenario is maintained by overzealous defeat: in the case of two equally plausible but conflicting arguments, only one is selected for further reasoning. The dismissal of the defeated argument, then, is justified by the rules of defeat laid down in the definitions.

I maintain that, among some type of arguments (called *generic inductive arguments*), sensible criteria of defeat cannot be defined. Thus instead of working at one state of the world at a time, it is much more realistic to branch into as many hypothetical worlds as needed, whenever there is lack of substantial evidence for plain defeat. In the literature, the option of maintaining concurrent worlds is known as, e.g., *multiple extensions* (Reiter, 1980), *multiple contexts* (De Kleer, 1986), *clusters of worlds* (Meyer and Van der Hoek, 1988), *frames of reference*, and *multiple belief spaces* (Martins and Shapiro, 1988).

This research was made possible by SION, and financed by NWO under contract number 612-316-019.

So the idea of branching worlds is in itself not new. The point with these systems is that not the proofs, but their supporting premises are defeasible. Therefore, these approaches do not stand so much for defeasible reasoning, as well as reasoning with defeasible information or, in more familiar terms, *belief revision*. Towards the end of the paper I argue that, besides belief revision, there is also something like *argument revision*.

I emphasize that this paper is generally non-constructive. Rather I try, by means of examples, counterexamples and rebuttals, to show which cases definitely lie beyond the scope of formal defeasible argumentation.

2. NOTATION AND BASIC CONCEPTS

It is not my intention to present a complete and definite system of defeasible reasoning. My principal aim is to discuss formal defeasible reasoning informally. This means that as many parameters will be left open for as long as possible.

However, I choose to make the following constants belong to the essentials of defeasible reasoning.

1. Atomic sentences p, q, r, \dots ; standard connectives $\neg, \supset, \wedge, \vee$; compound sentences ϕ, ψ, \dots and sets of sentences P, Q, \dots Upto this stage we are already committed to the vocabulary of propositional logic.

Furthermore a binary connective $>$, which informally stands for defeasible implication. $D\phi$ is a shorthand for true $> \phi$, and can be read as 'defeasibly ϕ ' or 'by default, ϕ '. (If one wants, D is able to capture something like *belief* or *defeasible knowledge*).

2. Defeasible proofs or *arguments* σ, τ, \dots ; sets of arguments Σ, \dots

In an obvious manner, arguments root in *premises* and end in *conclusions*; example: $\text{prem}(\sigma) = \{p, (p \vee q) > \neg r\}$, $\text{conc}(\tau) = \neg q$.

At this stage, many parameters of defeasible reasoning are still unspecified. Nevertheless, we can already show how a prototypical example of defeasible reasoning will be presented.

Example. (Whether penguins fly). Premises $P = \{p \supset b, p > \neg f, b > f\} \cup \{p\}$. Informally, p stands for penguins, b for birds, and f for flying creatures. The union tries to express that P can be thought of as available information, divided into *factual information* (or background knowledge) $\{p \supset b, p > \neg f, b > f\}$ and *contingent information* (or perceived knowledge) $\{p\}$.¹ Note that defeasible implications have their place within the factual information, in contrast with most approaches.²

¹The term *background knowledge* is suggested by Geffner and Pearl (1989). Instead of *premises*, Pollock (1987) uses the term *epistemic base*. In classical treatments on the theory of knowledge, the term *direct evidence* is often used.

²Notably Nute (1988), Poole (1988), and Reiter (1980). If we fol-

Defeasible implications mostly come as facts, and should then be treated as such.

The idea of splitting up P is that the observations will 'trigger' elements of the background knowledge. From a logical point of view, however, all members of P are of equal importance.

Obviously, there might be an argument σ , such that $\text{prem}(\sigma) = \{p, p > \neg f\}$ and $\text{conc}(\sigma) = \neg f$. Similarly, there might be an argument τ , such that $\text{prem}(\tau) = \{p, p \supset b, b > f\}$ and $\text{conc}(\tau) = f$. In that case, σ and τ would be *counterarguments* with conflicting conclusions f and $\neg f$. \square

We will gradually fill in further details in sections 3, 4, 5, 6, and 7.

3. CONSTRUCTION OF ARGUMENTS

Roughly speaking, arguments are considered here like proofs in tree-style *natural deduction* (Prawitz, 1965).

Example. (Elaboration of the preceding example).

$$\frac{\begin{array}{c} p \\ \hline p \quad p > \neg f \end{array}}{\neg f} \qquad \frac{\begin{array}{c} p \quad p \supset b \\ \hline b \end{array} \quad b > f}{\frac{\begin{array}{c} b \\ \hline f \end{array}}{p > f^{(1)}}} \qquad \frac{\begin{array}{c} [p]_1 \quad p \supset b \\ \hline b \end{array} \quad b > f}{\frac{\begin{array}{c} b \\ \hline f \end{array}}{p > f^{(1)}}}$$

The delicate nature of $>$, however, provides us with an additional restriction. Arguments with $>$ -introductions like the fourth one, will not be considered. The opportunity of making $>$ -introductions yields a fundamentally different argument system which deserves separate study.³

Arguments obviously have *subarguments*. If σ is a subargument of τ this will be denoted by $\sigma \sqsubseteq \tau$. The only arguments without proper subarguments are premises.

low the lines of Poole, the expression "if ϕ then normally ψ " would become defeasible knowledge of a strict implication $D(\phi \supset \psi)$, instead of strict knowledge of a defeasible implication $\phi > \psi$.

³Formally, a $>$ -introduction is strengthening the conclusive force of an argument, in contrast with other rules of inference. So arguments without $>$ -introductions may be expected to monotonically decrease in conclusive force. In the vocabulary of Prawitz (1965), our arguments are of *analytical type*. In the vocabulary of Pollock (1987), our arguments are of *linear type*.

4. THE NOTION OF POSSIBLE WORLD

Let us call a set of arguments a *possible world* if it (i) contains all subarguments of its elements, and (ii) contains no counterarguments.⁴ Terms like *scenario*, *hypothetical world*, *situation*, *state of affairs*, etc. would suit equally well. Note that a possible world stands for a *partial* and *unfinished* description of a state of the world.

Example. Let $\sigma = (p, p \supset q)/q$. Then the set of arguments $\Sigma = \{p, p \supset q, \neg q\}$ is a possible world, while $\Sigma \cup \{\sigma\}$ is not.

It is important to realize that my notion of possible world is not closed under any sort of argumentation whatsoever. Consequently, if some elements of a possible world are being continued, we obtain a new set of arguments, perhaps another possible world, but not necessarily so. Informally, in the process of reasoning we are supposed to reason from one possible-state-of-the-world to another.

Formally, we might speak of an *argumentation sequence*, written

$$\Sigma_1 \rightarrow \Sigma_2 \rightarrow \dots \rightarrow \Sigma_n \rightarrow \dots$$

starting at the epistemic base P with $\Sigma_1 = P$. If, given Σ_n , the next possible world is going to include counterarguments, then some elements of Σ_n need to be given up, preferably those which are falling short of conclusive force. In this way, we may declare an argument to be *in force* if it is contained eventually in every element of every argumentation sequence starting with P . So, loosely speaking, a good argument is one that proves the best for us in the long run.

Although the concept of argumentation sequence is a provisional one, it may be instructive to note that, besides nonmonotonicity at the *language level* (i.e. $\phi > \psi$ does not necessarily imply $(\phi \wedge \psi) > \psi$) and nonmonotonicity at the *inference level* (i.e. defeating arguments), we also have nonmonotonicity at the *knowledge level*. That is, an argument that is in some element of an argumentation sequence might be given up in a later stage, due to deliberation on arguments with relatively more conclusive force than the argument in question. This may sound straightforward, but there are theories in which every inferred formula becomes definitive.⁵

⁴ My notion of possible world has much in common with the notion of *argument structure* in the treatise of Lin and Shoham (1988). An argument structure satisfies, besides the conditions imposed on possible worlds, in addition the condition of *monotonic closedness*: it contains all deductive continuations of all its members. There is also resemblance to Nute's notion of *belief state*. A semantical notion like belief state is, like my notion of possible world, defined as a set of syntactic elements (Nute, 1988).

⁵ Reiter's default logic (1980) is about nonmonotonic reasoning, but still his default extension E is defined in terms of a strictly monotonically increasing sequence $(E_i)_{i=0}^{\infty}$. Obviously, this se-

5. MEANING OF ARGUMENTS

In decent logics, every sentence is given an unambiguous interpretation. Here I refuse to commit myself to one type of semantics. This means in particular that the connective $>$ will be interpreted as I find appropriate. Sometimes a maneuver that seems promising in one reading of $>$, might become less reasonable—or even wrong—in another reading of $>$. In this way, the interpretation of $>$ can be studied from different angles.

In the course of time there emerged two intentionally different interpretations of defeasible implication.

1. *Statistically based interpretation.* Sentences of the form $\phi > \psi$ are read as 'almost all ϕ -occasions are ψ -occasions'. With this interpretation, both ϕ and ψ must be $>$ -free.
2. *Generic interpretation.* Sentences of the form $\phi > \psi$ are read as ' ϕ is a good reason to infer ψ '. Here, the phrase "good reason" might be weakened to something like having a "legal alibi".

Note that, in a certain manner, reading (2) includes reading (1). This conceptual division in meaning is thoroughly discussed in chapter ten of (Pearl, 1988).⁶

Example. (Satisfiability and consistency of defeasible implications). The set $P = \{p \supset b, b > f, p \supset \neg f\} \cup \{p\}$ is satisfiable in a statistical reading of $>$, while it is definitely not consistent in a generic reading of $>$, witness the existence of two conflicting arguments rooting in P . \square

Example. (Chaining of defeasible implications). The inference $(p_1 > p_2, \dots, p_{n-1} > p_n) / (p_1 > p_n)$ will typically be judged as valid in a generic reading of $>$. Statistically, the rule is invalid as soon as $n \geq 3$. The information that almost all p_1 -occasions are p_2 -occasions and almost all p_2 -occasions are p_3 -occasions simply does not imply that almost all p_1 -occasions are p_3 -occasions.⁷ \square

Note that inferences which are (1)-valid are also (2)-

quence is not intended to represent some form of reasoning, but merely is a mathematical construction to define E . But, just because of this monotonicity, Reiter needs in the definition of E some form of 'look ahead' to prevent the inclusion of formulas that are given up in a later stage.

⁶ Besides Pearl, Reiter and Crisculo (1981) also have pleaded for a non-statistical reading of defeasible implications. Their terminology is: statistical connotation vs. *prototypical* connotation. For example, a prototypical mosquito tends to pass malaria although, statistically, most mosquito's in fact do not. Nute (1988): "probabilistic reasoning may be appropriate in some circumstances, but this should not be our only model for defeasible reasoning". Most researchers happen to be very keen on this issue.

⁷ Because we are dealing with a modified version of propositional logic, we are unable to denote classes of individuals. Therefore, the implication $b > f$ should not be read as "birds tend to fly," but (in a rather roundabout way) as "birds-occasions tend to be fly-occasions".

valid. For the same reason, (2)-consistency implies (1)-satisfiability.

6. THREE TYPES OF INFERENCE

In standard propositional logic there is but one type of inference, namely *deduction*. A deduction can be 'summarized' by introducing a material implication \supset . Conversely, \supset can be 'unfolded' by eliminating it via modus ponens. So \supset can be viewed as the object-symbol of logical deduction.

In a similar way, I choose to split argumentation into three types of inference, corresponding with the interpretation of \supset and the two different interpretations of $>$, respectively.

1. *Deduction*. All inferences in which the conclusion necessarily follows from the premises. For example, $(p, p \supset q) \supset q$, $(p \wedge q) \supset q$, et cetera, but also statistically valid inferences like $(p > q, q \supset r) \supset (p > r)$ and $((p \wedge r) > q, (p \wedge \neg r) > q) \supset (p > q)$.

It is known that propositional logic can be axiomatized in several ways. However, if $>$ appears only as an outmost connective, also the statistically based interpretation of $>$ has a *sound and complete axiomatization* by means of the following three rules of inference (cf. Geffner and Pearl, 1989):

- (i) weak transitivity: $(\phi > \psi, (\phi \wedge \psi) > \chi) \supset (\phi > \chi)$;
- (ii) weak monotonicity: $(\phi > \psi, \phi > \chi) \supset (\phi \wedge \psi) > \chi$;
- (iii) disjunctive antecedent: $(\phi > \chi, \psi > \chi) \supset (\phi \vee \psi) > \chi$.

The advantage of statistically valid inferences is that new defeasible implications can be obtained via the safest mode of inference, i.e. deduction.

2. *Statistically based induction*. With this type of inference, there is a statistical correlation between premises and conclusion.⁸ Example: $(p, p > q) \supset q$. Another typical example of statistically based induction is the set $P = ((p \wedge q) > r, p > \neg r) \cup \{p \wedge q\}$. This set statistically induces r , despite the presence of $p > \neg r$ in P .

In general, every statistically based conclusion ψ can be obtained as follows. First, P must deductively entail both ϕ and $\phi > \psi$. Moreover, there should be no ϕ' and ψ' such that P deductively entails both ϕ' and $\phi' > \psi'$, ϕ deductively follows from ϕ' , while ψ and ψ' are in contradiction. The additional condition ensures that the most specific reference class turns the scale (Loui, 1990).⁹

⁸With deductive inference, one might say that premises and conclusion are *categorically correlated*.

⁹The notion of 'most specific reference class' is properly subsumed by the more general notion of 'most specific explanation'. See, e.g., the paper of Loui (1990) for a detailed treatment on these issues.

3. *Generic induction*. Here, there is no statistical correlation. The only link between premises and conclusion is that they can be connected by some argument. The idea behind this type of inference is that generic induction stands for the *existence* of some argument, good or bad.¹⁰

These three types of inference should be understood as inclusive—not disjunct. So, for example, a statistically based inference is a generic inference at the same time.

Example. (Skew diamond).¹¹ Consider the set of premises $P = \{p \supset q, q > \neg s, p > r, r \supset s\} \cup \{p\}$. There are no deductive arguments for or against s (i.e. for $\neg s$). However, $\{p > r, p \supset s\}$ entails via proper deduction $p > s$ and hence, via statistically based induction, s . The only arguments against s are of generic inductive type. \square

Note that we refrain from concluding that the argument for s should defeat the argument against s . For the moment we are just interested in the types of the arguments involved.

Example. (Weak diamond).¹² Consider the set of premises $P = \{p > q, q > s, p > r, r > \neg s\} \cup \{p\}$. Again there are arguments for and against s , but now only of generic type. Consequently, there is no statistical correlation between P and s . Put differently, both $P \cup \{s\}$ and $P \cup \{\neg s\}$ are statistically satisfiable. (As easily can be verified.) \square

Example. (Disjunctive antecedents).¹³ In several defeasible logics,¹⁴ the set $P = \{p > r, q > r\} \cup \{p \vee q\}$ does not yield r as a defeasible conclusion. This is because, in those logics, both $p > r$ and $q > r$ must be triggered by specifically p or q , respectively. However, here r can be obtained by a statistically based inductive argument as follows. First perform the sound deduction $(p > r, q > r) \supset (p \vee q) > r$, and then perform the statistically based inference $(p \vee q, (p \vee q) > r) \supset r$. \square

Without the possibility of making generic inferences, arguments would already get stuck on an elementary level, since many inferences simply lack any statistical underpinning whatsoever. A generic inference tries to overcome this by surpassing statistical considerations.

¹⁰Besides deduction and statistically based induction, Pollock (1987) distinguishes more types of (basic) inference, viz. perception, memory and (Hume-like) induction. Apparently, his choice can be understood in the light of classical epistemology.

¹¹In the paper of Horty and Thomason (1988) presented as the problem whether native speakers of Pennsylvanian Dutch are born in America. There, p stands for 'native speaker of Pennsylvanian Dutch,' q for 'native speaker of German,' r for 'born in Pennsylvania,' and s for 'born in America'.

¹²Sometimes referred to in the literature as the Nixon diamond (Ginsberg, 1987).

¹³Principle of joint defeat (Pollock, 1987).

¹⁴In particular Reiter's default logic (1980).

Logically fundamental options like *chaining* and especially *nonmonotonicity* are only achievable by means of generic argumentation.

7. DEFEAT AND CONCLUSIVE FORCE OF ARGUMENTS

Depending on the type of inferences that were used, arguments may possess more or less conclusive force. As long as competing arguments clearly differ in conclusive force, it is also clear which of them should defeat the others.

1. *Deductive arguments defeat inductive arguments.* This is obvious.
2. *Statistically based inductive arguments defeat generic inductive arguments.* The underlying idea is that substantial statistical evidence carries more weight than just a line of reasoning.

To be sure, it may happen that a generic inductive argument is right where, at the same time, a statistically based inductive argument is wrong. But still this does not run counter to the principles on which statistically based arguments are preferred.

Conflicting deductive arguments refer to logically incompatible premises. This includes statistically unsatisfiable premises like $P = \{a > b, a > \neg b\}$.¹⁵ In that case some of these premises should be given up. This is the domain of *belief revision* which does not have my prime attention here.

Conflicting statistically based inductive arguments are the result of statistical biases (not of statistically unsatisfiable premises). Here, the most specific reference class must be taken to settle the matter.

Before we discuss defeat among conflicting generic inductive arguments, I wish to show that the ordering in conclusive force just mentioned is already capable of picking the right conclusion from non-trivial scenario's.

Example. (Whether Ralph fishes on Mother's day).¹⁶ Suppose that Ralph normally goes fishing on Sundays, but on the Sunday which is Mother's day, he typically visits his parents. Furthermore, in the spring of each leap year his parents take a vacation, so that they are not able to be visited.

Now the answers to the following three questions are of interest: (1) what does Ralph do on regular Sundays; (2)

¹⁵ There are algorithms to check whether a set of formulas is statistically satisfiable (see, e.g., Pearl, 1988). However, my interest does not so much go to logical or, in particular, statistical satisfiability of premises in the first place, but rather goes to the process of argument construction itself. This explains why my notion of possible world is not closed with respect to argumentation: hidden conflicts or inconsistencies should be brought to the fore by active argument construction, and not via some logical closure operator.

¹⁶ Konolige, 1988.

what does Ralph do on Mother's day; (3) what does Ralph do on Mother's day when his parents are taking their vacation?

Let $P = \{s > f, m > v, m \supset s, f \supset \neg v, l \supset \neg v\}$. Informally, s stands for 'it's Sunday,' f for 'Ralph is fishing,' m for 'it's Mother's day,' v for 'Ralph visits his parents,' and l for 'it's a leap year'.

1. On regular Sundays, Ralph should be fishing. Let us see what the ordering in conclusive force is telling us. The addition of s to P , yielding $P \cup \{s\}$, enables among others two relevant arguments, viz. $\sigma_1 = (s, s > f) / f$ and $\sigma_2 = (\sigma_1, f \supset \neg v) / \neg v$. (Both σ_1 and σ_2 are statistically based inductive arguments. That is not really important here because, at this stage, there are no competing counterarguments.) So, according to the ordering in conclusive force, Ralph goes fishing and, consequently, does not visit his parents.
2. On Mother's day, Ralph should visit his parents. The addition of m to $P \cup \{s\}$, yielding $P \cup \{s, m\}$, enables two statistically based arguments, viz. $\tau_1 = (m, m > v) / v$ and

$$\tau_2 = (\tau_1, [(f \supset \neg v) / \dots / (v \supset \neg f)]) / \neg f$$

where $(f \supset \neg v) / \dots / (v \supset \neg f)$ stands for a deductive argument turning $f \supset \neg v$ into $v \supset \neg f$. Moreover, $P \cup \{s, m\}$ enables, besides τ_1 and τ_2 , still σ_1 and σ_2 . Of all statistically based arguments, we should prefer the ones which are triggered by the most specific *reference class*, in this case m . Hence, σ_1 and σ_2 are defeated by τ_2 and τ_1 , respectively.

So, according to the two principles of defeat, Ralph is visiting his parents instead of fishing.

3. On Mother's day in a leap year, Ralph cannot visit his parents, and should therefore return to his fishing. In this case $P \cup \{s, m, l\}$ enables the deductive argument $p = (l, l \supset \neg v) / \neg v$. Hence, p defeats τ_1 and so Ralph does not visit his parents, which was to be expected. Much more interesting is it to see whether the system reinstates a scenario where Ralph is fishing again. A bit surprisingly, the system abstains. This is because there is no statistical support for f on basis of $m \wedge l$, since the implication $(m \wedge l) > f$ does not follow from P .¹⁷ (However, it is instructive to note

¹⁷ Counterexample: if the extension of a proposition is denoted by $[\phi]$, take, for every $n \geq 1$, $[s_n] = [f_n] \cup [m_n]$, $[f_n] = \{F_1, \dots, F_{n^2}\}$, $[m_n] = [v_n] \cup [l]$, $[v_n] = \{V_1, \dots, V_n\}$, and $[l] = \{L\}$. Capitals denote specific occasions. So, for example, $V_1 \in [s_n] \cup [m_n] \cup [v_n]$ is an occasion where Ralph is visiting his parents ($V_1 \in [v_n]$) on Mother's day ($V_1 \in [m_n]$). Now, for every $n \geq 1$, P is satisfied while $(m \wedge l) > f$ is not. (We have constructed infinitely many extensions, because a counterexample may not depend on the ratio of inclusion of both $s > f$ and $m > v$. So as n increases, the $>$'s are becoming tight fits, while $(m \wedge l) > f$ remains false.)

that, on basis of $s \wedge \neg v$, there is statistical support for f , (i.e. P deductively entails $s \wedge \neg v$ and $(s \wedge \neg v) > f$) while, on basis of m , there is statistical support for $\neg f$ (i.e. P deductively entails m and $m > \neg f$). Unfortunately, m and $s \wedge \neg v$ are logically incomparable with respect to P , so that there is no most specific reference class to settle the matter.)

Anyhow, the system does not tell whether Ralph goes fishing or not. Although this perhaps contravenes our intuition, this reservedness is correct because we have not told the system that every fourth Mother's day falls in a leap year (i.e. the premises do not represent that l is a substantial part of m). To be sure, if the extension of l in m cannot be taken arbitrary small, e.g. by demanding $(m \wedge \neg v) > l$, then $(m \wedge l) > f$ follows from P , indeed. Since $m \wedge l$ implies both m and $s \wedge \neg v$, it is the most specific reference class, and hence $m \wedge l$ is entitled, under these conditions, to settle f as a definite conclusion. \square

Despite the lack of statistically based evidence, item (3) still leaves us with a collection of conflicting generic inductive arguments. The next section is about this case.

8. DEFEAT AMONG GENERIC INDUCTIVE ARGUMENTS

At this stage, only the case of conflicting generic inductive arguments remains to be considered. Here, however, I will show that further inspection on the structure of arguments cannot establish a rational tie break. Still, many systems of defeasible reasoning have rules which peremptorily prefer one argument over the other to solve a conflict between competing generic inductive arguments.

A suitable example is the flood of new and altered logics designed to clarify or even to solve the famous *Yale shooting problem* (Hanks and McDermott, 1987). The Yale shooting problem revolves around the emergence of an anomalous argument supporting an unwanted conclusion. This anomalous argument is in force due to the rules of defeat as laid down in the definitions. So often, the first move is to designate the machinery of defeat of the system at issue as the culprit. After some tinkering to the system, the unwanted conclusion no longer turns up (in the Yale shooting scenario: Fred inevitably ceases to live).¹⁸ However, it looks very much like it that, with these approaches, the system has been adjusted merely to suit the scenario at issue. For many scenario's, including the Yale shooting, unique and unambiguous solutions simply do not exist.

As my logic is principally too weak to handle the Yale

¹⁸With tinkering I mean: introducing *abnormality predicates* (McCarthy, 1980), non-normal so-called *blocking defaults* (Reiter and Crisculo, 1981), introducing a *temporal priority-projection-preference ontology* (both Shoham and Lifschitz), etc.

shooting problem (it has no time component), I will take another example. This example shows how choices for specific kinds of defeat for systems of defeasible reasoning are usually justified.

Example. (Tuning the system). Let us consider a specific system of defeasible reasoning. After the criteria of defeat among arguments are formally settled, the system is usually tested against several case studies (benchmark problems) to see whether the right conclusions will emerge.

Now suppose that, with the results of the test cases at hand, we want to tune our system to remove anomalous conclusions (if any). Due to the considerations in section 7, it follows that if we want to make additional refinements in conclusive force, this can only be done within the class of generic inductive arguments.

Suppose that we want to establish this by preferring generic inductive arguments with superior specificity over those endowed with inferior specificity.¹⁹ After this alteration, we want to test whether specificity is a good criterion by setting our refined system again to the stock of test cases. Suppose the first test case concerns the well known student-adult-employed scenario.

Case study. (Whether young adults are employed).²⁰ Let $P = \{a > e, u > \neg e, (y \wedge a) > u\} \cup \{y \wedge a\}$. Informally, a stands for adults, u for university students, e for employed persons, and y for young people. The question is whether e or $\neg e$ should, on basis of P , be accepted as a warranted conclusion.

Now, at first it should be noted that deduction or statistically based induction cannot provide a definite answer. It follows that only a generic inductive argument may eventually turn the scale.

Suppose, for the sake of argument, that common knowledge compels us to attach the conclusion $\neg e$ to P (i.e. we all feel that, on basis of P , young adults generally don't work). Let us see what the system does. There are arguments σ and τ

$$\sigma: \frac{a \wedge y}{a} \quad a > e \qquad \tau: \frac{a \wedge y \quad (a \wedge y) > u}{u} \quad u > \neg e$$

respectively. As σ has a defeasible implication with a weakened antecedent (i.e. σ is weakening $a \wedge y$ to a), τ is

¹⁹This example does not particularly hinge on the criterion of specificity: any other additional criterion would do also. Specificity is nevertheless chosen because Loui remarks that "It's hard to imagine a system for selecting among competing defeasible conclusions that would not favor the conclusion with superior evidence, superior specificity, and 'equivalent directness'. (...) It's because specializing the antecedent strengthens the rules in such a way that the rule now dictates what should be done in the multiple extension situation." (Loui in response, 1987).

²⁰Pearl, 1989.

more specific than σ , so that—according to criteria of specificity— τ defeats σ . Hence, the system comes up with $\neg e$. We conclude that, as far as this case is concerned, the system runs parallel to common knowledge. This conclusion confirms our conviction that specificity is a sound criterion.

However, the following renaming of the preceding case study shows that the adapted system nevertheless remains ill-tuned.

Case study. (Whether bankrupt conservatives are selfish).²¹ Exchange a for c , y for b , e for s , and u for p . Informally, c stands for conservatives, b for bankrupt people, p for poor people, and s for selfish people. Consequently, the premise set of the previous case study P will become $P' = \{c > s, p > \neg s, (b \wedge c) > p\} \cup \{b \wedge c\}$. Hence the question is whether s or $\neg s$ holds.

Suppose, again for the sake of argument, that we generally feel that, on the basis of P , conservatives, whether they are bankrupt or not, poor or not, always tend to be selfish. So common knowledge compels us to attach s to P' . (Recall that there is no substantial evidence that provides a definite conclusion!) It will be clear that the system, as it previously supported $\neg e$, will now support $\neg s$. But $\neg s$ is unwanted as it contradicts common knowledge. Providing the system with other features of defeat will not help, since in that case the student-adult-employed scenario would be invalidated. □

From this example we learn in the first place that, thanks to the coarseness of our language, substantially different scenario's can be mapped onto a single set of formal premises. Conversely, the ‘inverse image’ of this set of premises shows too much variation to lift non-trivial principles of defeat out of it. We are bound to conclude that valid principles must, just because of their generality, be very weak. For the Yale shooting problem in particular, this means that sensible systems of defeasible reasoning, capable of capturing (but not particularly committed to) the Yale shooting scenario, refrain from pronouncing upon Fred’s aliveness. If not, then all variables in the Yale shooting scenario can be renamed in such a way that previously wanted conclusions now become unwanted. (I know that this has been observed before, but this is a sideline to the main issue).²²

²¹Poole, 1988.

²²Discussing why justification does not imply truth, Pappas and Swain (1978) write: “it is easy to imagine two cases which are identical in all respects, i.e. same evidence and same circumstances, except that the proposition justified is true in one case and false in the other”. Loui, in response to Hanks and McDermott (1987): “examples can be found at will that share the same syntactic structure (...), but do not seem to require the analogous solution.”

9. MULTIPLE POSSIBLE WORLDS

From the preceding considerations we learn that the reach of defeat has its limitations. In fact, an argument may be defeated if and only if there is a counterargument having superior conclusive force to reject the argument in question. Although generic inductive arguments do differ in conclusive force, these differences are too insignificant to be used in justifying criteria of defeat. Add to this that generic inductive arguments are deficient of substantial (read: statistical) conclusive force, and it is clear that we must conclude that there are no sensible principles for a general system of defeat among competing generic inductive arguments.

This conclusion has immediate consequences for defeasible reasoning in general. As generic inductive arguments all fall short of substantial conclusive force, no such argument has the right to defeat another. Because the bulk of defeasible reasoning goes beyond statistically based argumentation, many conflicting conclusions will have equal rights to remain in force. Thus, every collection of conflicting (but undefeated) generic inductive arguments will split an existing scenario into equally plausible alternatives. (Shuffling with principles like *collective defeat* or *sceptic inheritance* (Horty and Thomason, 1988) should be avoided since this would contravene the very nature of ampliative inference.)

Example. (Weak diamond). Consider the set of premises $P = \{p > q, q > s, p > r, r > \neg s\}$. P is a set of premises, but can also be seen as a set of arguments, say Σ_1 . The conclusions of these arguments do not contradict each other and, hence, Σ_1 forms by definition a possible world.

Suppose that P is updated with a new observation p . The addition of p , yielding $P \cup \{p\}$, enables the arguments $\sigma_1, \sigma_2, \tau_1$, and τ_2 .

$$\begin{array}{c} \sigma_1: \frac{p \quad p > q}{q} \quad q > s \\ \sigma_2: \frac{}{} \quad s \end{array} \quad \begin{array}{c} \tau_1: \frac{p \quad p > r}{r} \quad r > \neg s \\ \tau_2: \frac{}{} \quad \neg s \end{array}$$

Note that $\Sigma_1 \cup \{p\}$ are deductive arguments, σ_1 and τ_1 are statistically based inductive arguments, and σ_2 and τ_2 are generic inductive arguments. Moreover, σ_2 contradicts τ_2 . The addition of p to Σ_1 leads us to at least two new possible worlds, viz.

$$\Sigma_{2,1} = \Sigma_1 \cup \{p\} \cup \{\sigma_1, \tau_1\} \cup \{\sigma_2\}$$

and

$$\Sigma_{2,2} = \Sigma_1 \cup \{p\} \cup \{\sigma_1, \tau_1\} \cup \{\tau_2\}^{23}$$

Note that both worlds only have different generic inductive arguments.

²³At least two possible worlds, because more arguments may be constructed.

Suppose, finally, that $P \cup \{p\}$ is updated with a second observation $\neg r$. As $\neg r$ is in itself a deductive argument, it defeats τ_1 and, hence, τ_2 . It follows that $\Sigma_{2,1}$ as well as $\Sigma_{2,2}$ turn into the same possible world, viz. $\Sigma_3 = \Sigma_1 \cup \{p, \neg r\} \cup \{\sigma_1\} \cup \{\sigma_2\}$. \square

With this example, it is important to note that both updates p and $\neg r$ do not contradict P . Hence, knowledge revision is not at issue here, as the amount of knowledge merely increases during the supply of new information. But instead of knowledge revision, we do have something like *argument revision*. In this example, for instance, $\Sigma_{2,1}$ plus $\neg r$ becomes Σ_3 . So the introduction of $\neg r$ imposed a rational revision on the collection of outstanding arguments.

It is also important to note that an eventual split is never a necessary consequence of adding new information. Only pure argumentation can be the sole motive to make the split. In principle it is possible to add a potentially world-splitting observation to a possible world without ever performing the split in further argumentation.

Example. Again, consider Σ_1 . As we just have seen, the formula p is a potentially Σ_1 -splitting observation. However, a bad argumentation sequence might proceed

$$\begin{aligned}\Sigma_1 \cup \{p\} &\rightarrow \Sigma_1 \cup \{p, p \vee q\} \rightarrow \\ &\Sigma_1 \cup \{p, p \vee q, p \vee q \vee r\} \rightarrow \dots\end{aligned}$$

Clearly, such an argumentation sequence will never cause a split. \square

A sensible argumentation method will of course not form infinitely many disjunctions. Therefore, the question of how to define sensible argumentation methods which are giving every argument the right amount of resources, is another important issue. I will confine myself by referring to (Loui, 1990).

10. CONCLUSION

There are several ways to process new information. New information (in the form of an update or an observation) always enables, together with the factual information (in the form of background knowledge or laws), new arguments. As long as these arguments do not interfere, the amount of factual and derived knowledge (i.e. conclusions of arguments) increases monotonically.

However, as soon as arguments come in conflict with each other, there are several ways to restore 'epistemic quiescence'.

1. Vitiate the involved arguments by striking them at their roots. This typically is the sole option in deductive argumentation: the arguments themselves are perfectly sound, so a conflict among deductive arguments necessarily refers back to incompatible premises. Consequently, some part of the factual knowledge *itself* turns out to be untenable and must be given up. This is the well-known domain of belief revision (cf.,

e.g., Gärdenfors, 1988). Here, the central issue is to decide which premises should be removed first.

2. Vitiate the involved arguments by pruning them to a harmless size (i.e. take non-conflicting subarguments). This typically is not possible in deductive argumentation, since deductive arguments are necessarily generated by their premises. So pruning is reserved for arguments of inductive type only. Here, the issue is which arguments should be dealt with first.

I have argued that this is clear for certain combinations of arguments. Within the class of generic inductive arguments this is not clear. In case of pruning all conflicting generic inductive arguments by appealing to principles of collective defeat we should follow option (3).

3. No argument currently in force, is vitiated. To resolve the conflict, the acquired knowledge is branched into separate possible worlds. It must be possible that these possible worlds meet eventually.

I have tried to show that, besides the extensively explored option (1), there are more ways to cope with incoming information, of which option (2) is important, but not all-embracing. In other words, a general theory of defeat has limited capabilities. The issue of how option (3) should be formalized and should be combined with (1) and (2), is a problem of itself that needs further study.

In this light, current research needs to be mentioned. Despite the generally non-constructive character of this paper, some briefly mentioned notions need further formalization. This is now occupying our study.

Acknowledgements

My sincere thanks to Wiebe van der Hoek, John-Jules Meyer and Yao-Hua Tan for their readiness to discuss these issues with me. Finally many thanks to Irene Baker for her efforts in proofreading.

References

- GÄRDENFORS, P., *Knowledge in Flux: Modelling the dynamics of epistemic states*, MIT Press, London (1988).
- GEFFNER, H. AND J. PEARL, "A Framework for Reasoning with Defaults," report CSD-870058 R-94-III, Cognitive Systems Laboratory, University of California, Los Angeles CA (September 1989).
- GINSBERG, M.L., "Introduction," *Readings in Nonmonotonic Reasoning*, Los Altos, CA, pp. 1-23, Morgan Kaufmann (1987).
- HANKS, S. AND D. McDERMOTT, "Nonmonotonic Logic and Temporal Projection," *Artificial Intelligence* 33, pp. 379-412 (1987).
- HORTY, J.F. AND R.H. THOMASON, "Mixing Strict and Defeasible Inheritance," *Proceedings of the AAAI*, pp. 427-432 (1988).

- KONOLIGE, K., "Defeasible Argumentation in Reasoning about Events," *Methodologies for Intelligent Systems*, Amsterdam 3, pp. 380-390, Elsevier Science Publishing Co. (1988).
- LIFSCHITZ, V., "Formal Theories of Action," *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*, Los Altos, CA, pp. 35-58, Morgan Kaufmann Publishers, Inc. (1987).
- LIN, F. AND Y. SHOHAM, "Argument Systems: A Uniform Basis for Nonmonotonic Reasoning," Research Report, Stanford University, Department of Computer Science (1989).
- LOUI, R.P., "Response to Hanks and McDermott: Temporal Evolution of Beliefs and Beliefs about Temporal Evolution," *Cognitive Science* 11, pp. 283-297 (1987).
- LOUI, R.P., "Defeat Among Arguments: A System of Defeasible Inference," *Computational Intelligence* 3(2), pp. 100-106 (April 1987).
- LOUI, R.P., "Ampliative Inference, Computation, and Dialectic," in *AI and Philosophy*, ed. J. Pollock, MIT Press (1990).
- MARTINS, J.P. AND S.C. SHAPIRO, "A Model for Belief Revision," *Artificial Intelligence* 35, pp. 25-79 (1988).
- MCCARTHY, J., "Circumscription-A Form of Non-Monotonic Reasoning," *Artificial Intelligence* 13, pp. 27-39 (1980).
- MEYER, J.-J.CH. AND W. VAN DER HOEK, "Non-Monotonic Reasoning by Monotonic Means," report IR-171, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands (November 1988). An abridged version appears in *Proceedings of JELIA'90*, Springer LNCS.
- NUTE, D., "Defeasible Reasoning and Decision Support Systems," *Decision Support Systems* 4(1), pp. 97-110, North-Holland (1988).
- PAPPAS, G.S. AND M. SWAIN, *Essays on Knowledge and Justification*, Cornell University Press, London (1978).
- PEARL, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, Inc., Palo Alto CA (1988).
- POLLOCK, J.L., "Defeasible Reasoning," *Cognitive Science* 11, pp. 481-518 (1987).
- POOLE, D.L., "A Logical Framework for Default Reasoning," *Artificial Intelligence* 36, pp. 27-47 (1988).
- PRAWITZ, D., *Natural Deduction: A Proof-Theoretical Study*, Almqvist & Wiksell, Uppsala (1965).
- REITER, R., "A Logic for Default Reasoning," *Artificial Intelligence* 13, pp. 81-132 (1980).
- REITER, R. AND G. CRISCUOLO, "On Interacting Defaults," *Proceedings of the Int. Joint Conf. on A.I.*, pp. 270-276 (1981).
- SHOHAM, Y., *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*, Cambridge Mass., MIT Press Coop. (1988).

Qualitative Intercausal Relations, or Explaining "Explaining Away"

Michael P. Wellman
USAF Wright Laboratory
WL/AAA-1
Wright-Patterson AFB, OH 45433
wellman@wrdc.af.mil

Max Henrion
Rockwell International Science Center
444 High St, #400
Palo Alto, CA 94301
henrion@sumex-aim.stanford.edu

Abstract

Explaining away is a common pattern of reasoning in which the confirmation of one cause of an observed event reduces the need to invoke alternate causal hypotheses. However, the opposite of explaining away can also occur, in which the confirmation of one cause *increases* belief in another. We provide a general qualitative probabilistic analysis of intercausal reasoning, and identify the property of the interaction between the causes, *product synergy*, that determines which form of reasoning is appropriate. This extends Qualitative Probabilistic Networks (QPNs) to support qualitative intercausal inference about the directions of change in probabilistic belief. Because it eliminates some qualitative ambiguities arising in QPN reduction algorithms, product synergy complements the existing (additive) synergy concept. The process of explaining away also suggests the use of qualitative propagation-based algorithms analogous to message-passing techniques for numerical belief networks.

1 Introduction: Explaining Away

It is becoming increasingly clear that causal relationships play an important role in practical reasoning. One critical issue is the fundamental asymmetry between *predictive* (or causal) reasoning, that is, reasoning from cause to effect, and *diagnostic* (or evidential) reasoning, from effect to cause. Pearl [1988a] clearly illustrates this by the "sprinkler" example, depicted in Figure 1. Either *A*, "the sprinkler was on last night", or *B*, "it rained last night", could cause *C*, "the grass is wet." *C* could cause *E*, "the grass is cold and shiny," as well as *F*, "my shoes are wet."

Observation of one effect, *E*, cold and shiny grass, is evidence for *C*, wet grass, and hence predicts the other effect *F*, wet shoes. Confirmation of one cause, *A*, the

sprinkler, also leads to the expectation of *C*, wet grass. But it does *not* provide any evidence for the alternate cause *B*, rain. Suppose prior observation of wet grass had led to defeasible acceptance of rain. In a default reasoning scheme, confirmation of the sprinkler having been on should lead to a *retraction* of the rain hypothesis. In a probabilistic reasoning scheme, it should lead to a *reduced probability* of the rain hypothesis, even though the possibility of simultaneous sprinkling and rain is allowed.

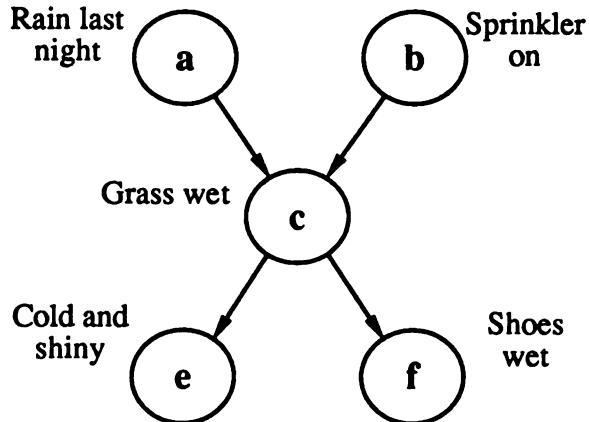


Figure 1: Causal diagram for the "sprinkler" example [Pearl, 1988a].

This is a common and intuitively compelling pattern of reasoning, called "explaining away" because one cause explains away the observed effect and so reduces the need to invoke other causes. This qualitative pattern of reasoning is entirely compatible with Bayesian inference when probabilistic influences reflect causal relationships [Henrion, 1987; Pearl, 1988a]. It is also the essence of Occam's razor: hypotheses should not be multiplied beyond what is necessary to account for the evidence.

Pearl [1988a] uses the revealed asymmetry of inference with respect to causal direction to argue for incorporating causal relations into default reasoning

schemes. Although inference rules implementing it have been well-studied [Geffner, 1990; Pearl, 1988a], precise and general conditions under which explaining away is valid or appropriate have not appeared in the literature.¹

Explaining away is an example of *intercausal inference* [Henrion, 1987], that is, reasoning between two causes with a common effect, in contrast with pure causal or pure evidential reasoning. While explaining away is often intuitively compelling, there are cases in which it appears inappropriate. Consider the following example illustrated by the causal model of Figure 2. You notice a newspaper headline about a well-known politician: "Senator Jones Killed in Car Accident." You idly wonder whether he might have been drunk. The headline gives no indication of whether he was at fault, or even whether he was a driver or passenger. You had no previous information about his driving or drinking habits, but you do recall hearing that about half of fatal car accidents in the United States are alcohol-related. Reading on, you find out that Jones was indeed the driver and no other vehicle was involved in the accident. How does this affect your belief that he had been drinking? Without knowledge of any accident, for most people the fact that the Senator was driving might *reduce* the suspicion that he had been drinking. But given the accident, many find the fact that he was the driver *increases* their suspicion. This would fit the all-too-common scenario of drinking and driving leading to a fatal accident. Note that this pattern of plausible reasoning is the *opposite* of explaining away: knowledge of a common effect renders a positive dependence between the causes, even though they were negatively dependent *a priori*.

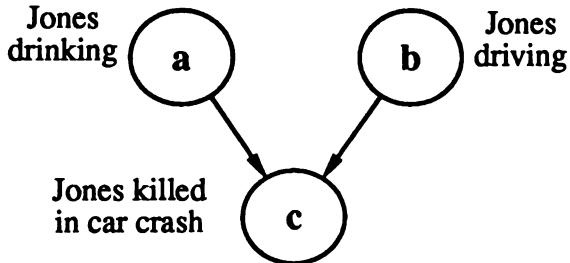


Figure 2: The drinking-and-driving example. When explaining away fails.

The goal of this paper is to provide a general analysis of intercausal reasoning accounting for both of the illustrated patterns of reasoning, and making precise the conditions distinguishing between them. The

¹Pearl provides these conditions for the special case of linear/Gaussian models [Pearl, 1988b, page 351] (which is illustrative). Explaining away can also be justified probabilistically by ϵ -semantics [Geffner, 1988], but these do not capture the full range of situations where such inference is appropriate.

problem has to do with degrees of belief and strengths of argument, which suggests that a purely deterministic formulation would be inadequate. Our choice of a probabilistic approach reflects the uncertainty central to causal explanation tasks, and is supported by the observation that "explaining away" is a natural consequence of some Bayesian causal models.

While the probabilistic formulation refers to quantitative degrees of belief, it does not necessarily require precise numerical probabilities for application. Indeed our analysis is qualitative, concerning the direction of probabilistic dependencies, but not their magnitude. Our premise is that the critical distinctions correspond to intuitive categories of interaction among causes, and that further precision would be impractical or less convenient and, for many purposes, unnecessary. This position is supported by the observation that common vocabulary includes numerous qualitative concepts of causal interaction. For example, we often say that causal factors act independently or synergistically, that one cause (a "gating condition") enables or inhibits another, or that a set of available inputs are complementary or substitutable with each other. Rain and the sprinkler independently cause wet grass, while drinking amplifies the causal relation between driving and car accidents.

We formalize these concepts using the Qualitative Probabilistic Network (QPN) representation [Wellman, 1990b], an abstraction of Bayesian networks. The analysis of intercausal reasoning extends this formalism by introducing new qualitative characterizations of causal interactions, complementary with the existing QPN *synergy* relations.

We have motivated our analysis by the attempt to replicate patterns of reasoning that people find intuitively compelling. There is, of course, copious empirical evidence that Bayesian reasoning is not an accurate model of our human intuitive reasoning, at least in quantitative terms. But, in its qualitative form, Bayesian reasoning does seem compatible with the patterns exhibited by these cases. Indeed, to our knowledge, it is the only formal approach capturing the intuitive concept of explaining away. Thus the qualitative version has appeal both as a normative and a descriptive model of reasoning.

In the remainder of this paper, we present a formal analysis of qualitative intercausal relations. The next section examines intercausal reasoning with a deterministic causal relationships. After reviewing the notion of qualitative probabilistic influence, we analyse intercausal reasoning with uncertain causal influences, and provide the condition for explaining away to occur. We generalise this to handle prior intercausal relationships, and for partial evidence on the effect. We compare the new notions of synergy with earlier concepts from the literature, and examine the application of these new results on intercausal reasoning in infer-

ence algorithms. Finally, we present a richer view of Occam's razor suggested by intercausal relations.

2 Deterministic Intercausal Relations

Although the explaining away phenomenon is significant primarily for reasoning under uncertainty, it is instructive to start with a simple analysis of the deterministic propositional case.² Let a , b , and c be propositional variables, and A and \bar{A} denote the literals where $a = \text{true}$ and $a = \text{false}$, respectively. We interpret the assertions $\text{causes}(A, C)$ and $\text{causes}(\bar{B}, C)$ to require at least that c depends on both a and b in a strict positive fashion. There are only two binary functions of a and b satisfying this property: OR and AND. The former justifies explaining away. If $c = \text{OR}(a, b)$, then if C holds, evidence for A tends to decrease support for B . Because \bar{B} is possible only when A holds, belief in B given A is necessarily less than its belief given \bar{A} . When $c = \text{AND}(a, b)$, however, there can be no explaining away given C because the observation immediately establishes both A and B .

The condition for explaining away is also quite simple for the case where the effect, c , is a real-valued function of a and b , increasing in both causal variables. If c is differentiable, we can derive the intercausal relation between a and b by applying the chain rule for partial derivatives:

$$\frac{\partial b}{\partial a}(a, c) = -\frac{\frac{\partial c}{\partial a}(a, b)}{\frac{\partial c}{\partial b}(a, b)}.$$

Since both partial derivatives on the right-hand side are positive by assumption, an increase in a requires a concomitant decrease in b (i.e., a "explains away" b) for any fixed value (e.g., an observation) of c .³ The fact that the intercausal relation is independent of the interaction between a and b in causing c suggests that explaining away is not a very interesting phenomenon for the case of invertible deterministic functions. The problem of explaining away uncertain causal hypotheses presents more subtle issues and requires a richer vocabulary of intercausal concepts.

3 Qualitative Probabilistic Networks

Our analysis of intercausal inference under uncertainty is based on the *QPN* formalism for qualitative probabilistic reasoning [Wellman, 1990b]. In a *qualitative probabilistic network*, variables are represented as nodes in a graph, with directed edges defining probabilistic relationships. As in Bayesian networks [Pearl,

²In fact, diagnostic and intercausal reasoning may be uncertain even when the causal structure is deterministic.

³For a general discussion of transformation rules on qualitative functional relationships, see the work of Michelena and Agogino [1989] on deterministic monotonic influence diagrams.

1988b] and other graphical schemes, connectedness in the graph represents the dependency structure of its underlying probability distribution [Pearl et al., 1989]. However, rather than specify the distribution precisely with numeric probability tables, QPNs merely constrain the conditional probabilities using qualitative influences. This qualitative approach bears a relationship to numeric probabilistic networks analogous to the relation of qualitative physics schemes to ordinary differential equations.

Associated with each edge is a sign, $\delta \in \{+, -, 0, ?\}$, denoting the direction of *qualitative influence* between nodes. Figure 3 depicts an example qualitative probabilistic network representing beliefs about the health of a friend. Event A , that our friend has a cold, increases⁴ the probability of C , that she is sneezing. Event B , that she has an allergic reaction, also increases this probability. On the other hand, event F , that she recently took an antihistamine tablet, reduces the probability of sneezing, that is, f has a negative influence on c . Event D , that our friend is allergic to cats, increases the probability of B , the allergic reaction, as does E , that a cat is present. (Whereas for ease of exposition the variables in our examples are all propositional, the definition of qualitative influence below, along with most other definitions and theorems, apply equally to multivalent discrete and continuous variables.)

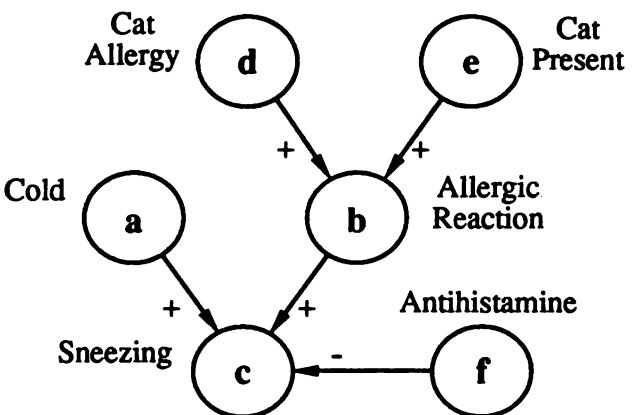


Figure 3: An example qualitative probabilistic network. Arrows labeled "+" and "-" denote positive and negative causal influences.

For the general definition of qualitative influences, consider a QPN with a directed edge from a to c , and optionally some other variables, collectively denoted z , with links to c . In Figure 3, for example, z would comprise b and f . This structure dictates that the probability distribution for c can be specified conditionally on a and z .

⁴We use terms such as *increase* and *decrease* in the non-strict sense, unless explicitly stated otherwise.

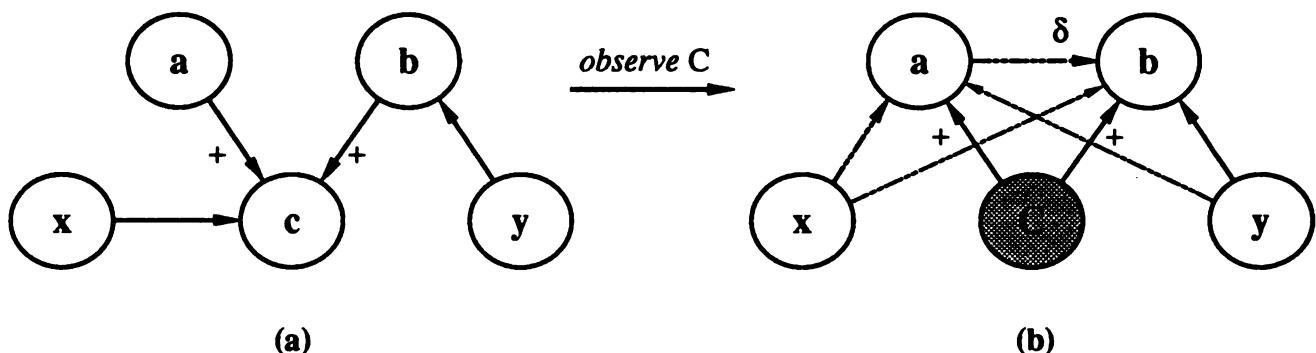


Figure 4: The schematic QPN transformation for intercausal inference. The qualitative intercausal relation, δ , of a on b upon observation of C indicates whether explaining away occurs.

Definition 1 (qualitative influence) We say that a positively influences c , written $S^+(a, c)$, iff for all values $a_1 > a_2$, c_0 , and x ,

$$\Pr(c \geq c_0 | a_1 x) \geq \Pr(c \geq c_0 | a_2 x).$$

An equivalent condition is that the probability density function⁵ for a given c and x , $f_a(\cdot|cx)$, obeys the *monotone likelihood ratio property*:

$$\frac{f_a(a_1|c_1x)}{f_a(a_1|c_2x)} \geq \frac{f_a(a_2|c_1x)}{f_a(a_2|c_2x)}, \quad (1)$$

for all $a_1 > a_2$, $c_1 > c_2$, and z . This property ensures that increasing a increases the expected value of c .⁶ Replacing \geq in (1) with \leq and $=$ yields the conditions for S^- and S^0 , respectively. S^0 means a and c are independent for all values of z . $S^?$ always holds, and is used to indicate that the relationship may be ambiguous.

The probabilistic definition for qualitative influences justifies simple inference rules based on graphical transformations of the network. For example, in Figure 3, given $S^+(e, b)$, $S^+(b, c)$, and no direct link from e to c (implicitly S^0), we can reduce b from the network and replace the two-link chain with a new qualitative influence, $S^+(e, c)$. In other words, a cat being present increases the probability of sneezing. In general, chains of influences combine by multiplication of the qualitative signs. Directed links in the network can also be reversed, with new qualitative influences computed from the old via a qualitative version of Bayes's rule. For example, sneezing, C , (in the absence of other evidence information) increases the probabilities of cold, A , and allergic reaction, B , and decreases

⁵For the discrete case, the appropriate probability mass function should be substituted.

⁶In writing these ratios, we assume that all conditional probability terms are well-defined and nonzero. These assumptions could be relaxed at the expense of explicatory complexity. For further discussion of these probabilistic inequalities and their implications, see [Milgrom, 1981; Wellman, 1990b].

the probability that our friend had recently taken an antihistamine, F . Each of these operations represents a valid transformation of the network with respect to the interpretation of qualitative influences as probabilistic constraints.

4 Probabilistic Intercausal Relations

Figure 3 also exhibits the phenomenon of "explaining away." Suppose we observe our friend sneezing, C , raising the probability of her having a cold, A , and the probability of allergic reaction, B . If we know she is allergic to cats, D , and learn that a cat is present, E , lending strong confirmation of the allergic reaction B , this explains away the sneezing, and so reduces the probability of the cold, A .

This process of intercausal inference can be cast as transformation of a causal graph or QPN, illustrated in general form in Figure 4. Again, a and b are causes of c . For generality, we allow that there may be other causes of c (collectively represented by x), and that a and b in turn may have causal antecedents (b 's are collectively labeled y ; a 's do not figure in the example). Figure 4a depicts this initial situation. Note that because their only connecting path is via their direct links to c , a and b are marginally independent in this model, although they are conditionally dependent given c .

The basic “explaining away” scenario starts with an observation of the effect variable to be explained, c . Suppose that c is propositional, and that the observed value is C . To represent observation in a probabilistic network, we instantiate the observed node and modify the dependency structure in the graph so that the nodes of interest become conditional on the observation. This entails reversing the links from a and b to c , as shown in Figure 4b. The signs on the reversed links remain positive, indicating that observing C increases the probability of higher values of a and b . In addition, the reversals introduce a new *intercausal* link between a and b , accounting for the fact that the variables be-

come dependent upon observing C . The "explaining away" pattern is characterized exactly by the negativity of this intercausal influence. For propositional a and b , the relation $S^-(a, b)$ in the graph of Figure 4b would mean that

$$\Pr(B|ACxz) \leq \Pr(B|Cz) \leq \Pr(B|\bar{A}Cz),$$

and hence belief in A decreases belief in B .

Even if we knew that the signs on the original links from a to c and b to c were positive, the sign on this new intercausal link would be ambiguous; that is, it would not be determinable from the given qualitative information. The question is what condition on the causal combination of a and b would enable us to derive a negative intercausal influence upon observing C ? In other words, we are after a general characterization of the criterion for explaining away. Moreover, we desire a local, qualitative condition, ideally one that corresponds to intuitive concepts of causal interaction. The product synergy property meets these requirements.

Definition 2 (product synergy) Let a , b , and x be the predecessors of c in a QPN. Variables a and b exhibit negative product synergy with respect to a particular value c_0 of c , written $X^-(\{a, b\}, c_0)$, if for all $a_1 > a_2$, $b_1 > b_2$, and x ,

$$\begin{aligned} \Pr(c_0|a_1b_1x)\Pr(c_0|a_2b_2x) &\leq \\ \Pr(c_0|a_1b_2x)\Pr(c_0|a_2b_1x). \end{aligned} \quad (2)$$

Positive product synergy, X^+ , and zero product synergy, X^0 , are defined by substituting \geq and $=$, respectively for \leq .⁷ We say that the qualitative relation is strict if its defining inequality—in this case (2)—holds strictly for some instances.⁸ To handle the case of continuous as well as discrete c , we can rewrite (2) as a relation on ratios of probability densities or masses:

$$\frac{f_c(c_0|a_1b_1x)}{f_c(c_0|a_1b_2x)} \leq \frac{f_c(c_0|a_2b_1x)}{f_c(c_0|a_2b_2x)}. \quad (3)$$

Thus, negative product synergy requires that the proportional increase in the probability of c_0 upon raising b is smaller for higher values of a . Hence, the causal contribution of a given variable is greatest when it is the only active (high-valued) cause. It is this type of interaction that underlies explaining away.

Theorem 1 (explaining away) Let a , b , and x be the predecessors of c such that a and b are conditionally

⁷The X^+ relation is equivalent to the condition that the probability function $\Pr(C|ab)$ be totally positive of degree two [Karlin, 1968] with respect to its arguments a and b . Whereas total positivity has been studied in some probabilistic contexts [Karlin and Rinott, 1980], the concept has not been applied in the same manner, and the results below—to our knowledge—are new.

⁸Although most of our results in this paper are presented only for the nonstrict case, extending them to strict relations is generally straightforward.

independent given y , that is, $S^0(a, b)$. A necessary and sufficient condition for $S^-(a, b)$ upon observation of c_0 is negative product synergy, $X^-(\{a, b\}, c_0)$.

This result follows directly from application of Bayes's rule reversing the dependence of c on b .⁹ The theorem is also valid with either "+" or "0" substituted for "—" in both the intercausal influence S^- and synergy X^- .

To illustrate this, let us reconsider the two examples of explaining away. In Figure 1, there is a negative intercausal relation between the sprinkler, A , and rain, B , given their common effect, wet grass, C . Figure 3 displays a corresponding negative intercausal relation between the cold and allergic reaction given their common effect, sneezing. According to Theorem 1, this kind of relationship is appropriate if and only if we believe that negative product synergy holds in each of these cases. In other words, our beliefs about the causal effects must satisfy the following relation, the propositional form of (3),

$$\frac{\Pr(C|AB)}{\Pr(C|A\bar{B})} \leq \frac{\Pr(C|\bar{A}B)}{\Pr(C|\bar{A}\bar{B})}. \quad (4)$$

In words, the proportional increase in probability of C , wet grass, due to learning B , rain, is smaller given A , sprinkling, than given \bar{A} , no sprinkling. Or, the proportional increase in probability of sneezing due to learning there is an allergy is less given a cold than given no cold. Both of these conditions seem eminently plausible—given one cause is present, the incremental effect of the second cause is less than if the first were absent.

If negative product synergy does not seem immediately intuitively compelling, one can also derive it from the "leaky noisy OR" [Henrion, 1989; Pearl, 1988b], a plausible model for either situation. The noisy OR dictates that each of the two causes may be sufficient alone to cause the effect, and that the causal mechanisms are independent. That is, the event that A is sufficient to cause C is independent of whether B is sufficient. The "leakiness" allows that, even if neither A nor B occur, C may occur for other unspecified reasons (a "leak" L). Let p_a and p_b be the probabilities that A or B alone, respectively, are sufficient to cause C , and p_0 be the probability that a leak causes C , without A or B . Thus we have

$$\begin{aligned} \Pr(C|\bar{A}\bar{B}) &= p_0, \\ \Pr(C|A\bar{B}) &= 1 - (1 - p_0)(1 - p_a), \\ \Pr(C|\bar{A}B) &= 1 - (1 - p_0)(1 - p_b), \text{ and} \\ \Pr(C|AB) &= 1 - (1 - p_0)(1 - p_a)(1 - p_b). \end{aligned}$$

Inserting these into the expression for negative product

⁹Complete proofs of this and other results are provided in the appendix. A propositional version of Theorem 1 appears in [Henrion and Drusdzel, 1990].

synergy (4), we obtain

$$\frac{1 - (1 - p_0)(1 - p_a)(1 - p_b)}{1 - (1 - p_0)(1 - p_a)} \leq \frac{1 - (1 - p_0)(1 - p_b)}{p_0}.$$

This is valid for all p_0 , p_a , and p_b in $[0, 1]$. In other words, the leaky noisy OR relation implies negative product synergy, and so leads to explaining away, or a negative intercausal inference. This result generalizes straightforwardly to cases with more than two causal variables. In contrast, “noisy NOR” models—where causes lead to the *negation* of the effect—exhibit zero product synergy.

Now let us reconsider examples where explaining away does not seem to apply. The drinking and driving Senator from Figure 2 is one such instance. The case from Figure 3 of the two causes of an allergic reaction is another. Given that an allergic reaction, B , is observed, knowledge that our friend is allergic to cats, D , seems to increase the probability that a cat is present, E , and vice versa. There is a positive intercausal relationship between D and E , given B . According to the positive version of Theorem 1, this occurs iff positive product synergy applies, that is

$$\frac{\Pr(B|DE)}{\Pr(B|D\bar{E})} \geq \frac{\Pr(B|\bar{D}E)}{\Pr(B|\bar{D}\bar{E})}. \quad (5)$$

For our example, this holds when the proportional increase in probability of an allergic reaction, due to the cat being present, is greater given our friend is allergic to cats, then if she is not. This is evident, given that the cat would have only indirect effects, if any, if she is not allergic to cats. Therefore, the right-hand side of (5) would be at or near unity, while the left-hand side would be significantly larger.

5 Dependent Causes

Application of Theorem 1 requires that causes a and b be marginally independent. We can generalize the result, so long as any prior dependence between the causes is in the same direction as the corresponding intercausal effect of observing their common finding:

Theorem 2 *Let a , b , and x be the predecessors of c . A sufficient condition for $S^{\delta}(a, b)$ upon observation of c_0 is an influence $S^{\delta}(a, b)$ prior to the observation plus a product synergy relation, $X^{\delta}(\{a, b\}, c_0)$, of compatible sign.*

For example, suppose we know our neighbor habitually listens to weather reports and turns off the sprinkler when rain is forecast. This negative prior relation between the two causes is in the same direction as the intercausal relation, and hence the tendency of the sprinkler to explain away the rain hypothesis is only strengthened.

On the other hand, suppose we believe in Murphy, the perverse raingod who likes to make it rain soon after a

sprinkler has been used. This induces a positive prior dependence between the causes, rain and sprinkler. In this case the intercausal relationship after observing wet grass becomes ambiguous and cannot be determined by purely qualitative analysis.

6 Related Synergy Concepts

Although it precisely characterizes the validity of intercausal reasoning, product synergy is not the only qualitatively significant condition on causal interaction. A similar concept, called *additive synergy*, has previously been defined for QPNs [Wellman, 1990b].

Definition 3 (additive synergy) *Variables a and b exhibit negative additive synergy with respect to variable c , written $Y^-(\{a, b\}, c)$, if for all $a_1 > a_2$, $b_1 > b_2$, x , and c_0 ,*

$$\begin{aligned} \Pr(c \geq c_0 | a_1 b_1 x) + \Pr(c \geq c_0 | a_2 b_2 x) &\leq \\ \Pr(c \geq c_0 | a_1 b_2 x) + \Pr(c \geq c_0 | a_2 b_1 x). \end{aligned} \quad (6)$$

Positive additive synergy, Y^+ , and zero additive synergy, Y^0 , are defined similarly, substituting \geq and $=$, respectively, for \leq . An important difference between additive and product synergy is that the former is defined with respect to the variable c rather than a particular value c_0 —that is, the additive synergy condition holds for all values of c . The disparity is due to the distinct roles of these relations in qualitative probabilistic inference. Note, however, that when c is a propositional variable, $Y^{\delta}(\{a, b\}, c)$ is identical to $X^{\delta}(\{a, b\}, C)$ except in substituting addition for multiplication in (2) (or differences for quotients in (4)).

The additive synergy condition possesses several desirable qualities. It is convenient for graphical inference because it combines naturally with qualitative influences for predictive reasoning (e.g., $Y^+(\{a, b\}, c) \wedge S^+(c, d) \Rightarrow Y^+(\{a, b\}, d)$ if there are no direct influences from a or b to d). Most importantly, additive synergy on utility entails the monotone decision property, enabling qualitative reasoning about optimal decisions (see Wellman [1990a; 1990b] for details). But Y^{δ} is neither necessary nor sufficient for intercausal reasoning. Conversely, the intercausal relation X^{δ} does not share all of the desirable properties of additive synergy.

However, if both of the individual influences of each cause on the effect have unambiguous signs (+ or -), there are entailment relationships between propositional product synergy and additive synergy. These depend on whether the signs of qualitative influence agree or disagree. Let c be a propositional variable, with influences $S^{\delta_a}(a, c)$ and $S^{\delta_b}(b, c)$.

Theorem 3 *If $\delta_a = \delta_b$, then*

$$\begin{aligned} X^+(\{a, b\}, C) &\Rightarrow Y^+(\{a, b\}, c), \text{ and} \\ Y^-(\{a, b\}, c) &\Rightarrow X^-(\{a, b\}, C). \end{aligned}$$

Theorem 4 If $\delta_a = -\delta_b$, then

$$\begin{aligned} X^-(\{a, b\}, C) &\Rightarrow Y^-(\{a, b\}, c), \text{ and} \\ Y^+(\{a, b\}, c) &\Rightarrow X^+(\{a, b\}, C). \end{aligned}$$

These results provide sufficient, but not necessary conditions for additive and product synergy to coincide.¹⁰ For example, we have already shown (Section 4) that the noisy OR model exhibits negative product synergy on C , regardless of the parameters. Noisy ORs are also additively sub-synergistic (which can be verified directly [Wellman, 1990b]), although they do not satisfy the antecedent of Theorem 8. However, by Theorem 7 the product sub-synergy follows from the additive.

Given a product synergy relation for a and b with respect to C , what can we say about the variables' interaction given \bar{C} ? For the noisy OR model, it turns out that there is no interaction, $X^0(\{a, b\}, \bar{C})$. In general, however, it is possible that product synergy holds in the same or opposite directions (perhaps even strictly) given C and \bar{C} , respectively. One sufficient condition for product synergy to hold for both C and \bar{C} is that the product and additive synergy relations disagree.

Theorem 5

$$X^\delta(\{a, b\}, C) \wedge Y^{-\delta}(\{a, b\}, c) \Rightarrow X^\delta(\{a, b\}, \bar{C}).$$

Table 1 summarizes the possible product synergy relations for \bar{C} given product synergy for C and additive synergy for c . The tabulated results presume that a and b are propositional and the sole causal antecedents of c .

$X^\delta(\bar{C})$	$X^-(C)$	$X^+(C)$
$Y^-[c]$	+	+
$Y^+[c]$	-	0

Table 1: The possible combinations of additive synergy and product synergy for C and \bar{C} . The shaded cell covers noisy OR distributions. Regions adjacent to the horizontal or vertical boundaries between quadrants are consistent with zero additive or product synergy, respectively.

By our previous results, the noisy OR relation is represented in synergy space by the shaded cell in Table 1,

¹⁰Theorems 7 and 8 in the appendix provide slightly more general conditions. However, complete (necessary and sufficient) conditions cannot be provided purely in terms of ordinal comparisons on pairs of $\Pr(c \geq c_0 | a; b; z)$ terms, which are the only relations constrained by qualitative influences.

corresponding to negative product and additive synergy, and zero product synergy for its negation. In fact, this pattern of synergy serves as a new qualitative definition of noisy OR, generalized slightly to admit some negative values for p_a and p_b (subject to the constraint that the noisy OR formulas of Section 4 produce valid probabilities).

Note that, by Theorem 3 (respectively 4), the upper-right (lower-left) quadrant of the table is empty when a and b influence c in the same (opposite) direction. Theorems 7 and 8 in the appendix present slightly more general conditions under which these quadrants can be ruled out on the basis of the individual relations of a and b on c . However, none of the regions of the table are null in general. This indicates that each of the synergy relations adds nontrivial information about the qualitative character of the intercausal relation. More significantly, each of these qualitative properties has important problem-solving implications. For example, Y^δ has decision-theoretic consequences regarding the optimal assignment to control variables (see [Wellman, 1990a]), while explaining away hinges on X^δ . The choice of synergy relations to apply in a particular reasoning problem depends on the available causal domain knowledge and the objectives of inference. For many applications, it would be advantageous to maintain all of the various qualitative synergy relations.

7 Indirect Evidence

Theorem 1 also presumes that the effect variable, c , is directly observed. Can we generalize this to situations where we have only indirect evidence for c ?

Suppose we observe the value of variable e , an effect of c . For example, in the sprinkler model, we might observe E , cold and shiny grass. To determine the intercausal implications of this observation, we investigate the interaction relation of a and b on e when c is factored out. This situation is depicted in Figure 5.

The following result establishes (for the propositional case) that evidence positively related to the effect¹¹ maintains intercausal relations, under a few additional assumptions. In particular, cold shiny grass is sufficient for the sprinkler to explain away rain.

Theorem 6 Suppose $X^{\delta_1}(\{a, b\}, C)$, $Y^{\delta_2}(\{a, b\}, c)$, and $S^{\delta_3}(c, e)$ hold in a network with no direct links from a or b to e . Then $X^{\delta_1}(\{a, b\}, E)$ holds in the network after reducing c if either:

1. $\delta_1 = \delta_2$ and $\delta_3 = +$, or
2. $\delta_1 = -\delta_2$ and $\delta_3 = -$.

Notice that both product and additive synergy are required to propagate product synergy across qualita-

¹¹Also called "favorable" evidence [Milgrom, 1981].

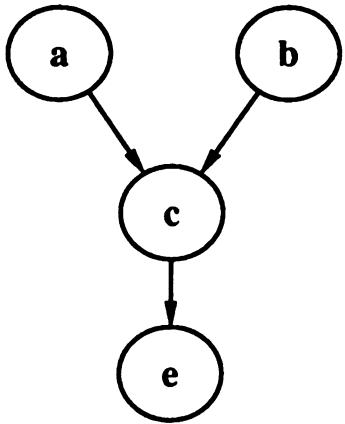


Figure 5: Two causes a and b , with partial evidence, e , for their common effect, c .

tive influences. According to Theorems 3 and 4, the synergy relations often coincide. For example, in the sprinkler case, the influences of a and b on c , and of c on e are all positive and there is negative additive synergy on c , $Y^-(\{a, b\}, c)$, so there will also be negative product synergy on e , $X^-(\{a, b\}, E)$. Thus observing indirect evidence E , grass is cold and shiny, induces explaining away (a negative intercausal influence) between a , rain, and b , sprinkler.

Under certain circumstances, we can generalize Theorem 6 to the case of non-propositional c . In essence, product synergy extends from c_0 to e_0 as long as e_0 supports c_0 but does not distinguish among $c \neq c_0$.¹² For propositional c , it matters only whether the observed value e_0 was more likely given C than \bar{C} .

8 Graphical Inference

It is a straightforward matter to incorporate product synergy into existing QPN inference algorithms [Wellman, 1990c]. The results on intercausal reasoning and indirect evidence correspond directly to QPN transformation rules. Moreover, the two forms of synergy are complementary: both are required for evidence extension (Theorem 6), and the intercausal inferences justified by product synergy can sometimes resolve qualitative ambiguity arising in basic QPNs.

One significant difference between product synergy and the other qualitative relations is that X^δ refers to a particular value of the variable rather than the entire scale. For *evidence propagation* [Shachter, 1989]—the operation of accounting for an observation of one of the nodes in the network—only the synergy relation

¹²To establish this, we divide c into values for which X^δ holds (C) and those for which it does not (\bar{C}), then apply the previous theorem. In the process, we must be careful that the division does not invalidate the conditional independence of a and b from e given c .

corresponding to the observed value is relevant. For the other operations, we must maintain the product synergies associated with each value as the network is transformed. For example, the evidential propagation rule of Section 7 demonstrates how synergy for particular values of a variable can be transferred to qualitatively related values of a successor variable.

When an evidence variable is directly observed, the preferred QPN operation is evidence propagation. Sometimes, however, it is necessary for chaining predictive or evidential inference to reverse the orientation of a link from b to c even though c is not observed. In such cases, deriving an unambiguous intercausal relation on b from some other causal predecessor, a , of c requires that product synergy $X^\delta(\{a, b\}, c_0)$ hold for all possible values c_0 of c . If δ depends on the particular value c_0 , the intercausal relation is ambiguous. And note that additive synergy does not resolve the ambiguity, except as it constrains the values of product synergy for different values of the variable (e.g., as in Table 1).

Upon reversing its link to c , b gains both a and c as causal predecessors. Given the original qualitative model, what can we say about the interaction of a and c on b in the transformed situation? It turns out that the qualitative relations in the initial configuration are not sufficient to determine the new qualitative synergy (either product or additive) holding after reversal.

The process of explaining away also suggests a quite different approach to inference in QPNs, based on *qualitative propagation* [Henrion and Drusdzel, 1990]. Instead of manipulating the topology of the network by reducing nodes and reversing links (as in Figure 4), propagation methods use the network structure to pass messages and combine qualitative values. Just as for numeric probabilistic networks, valid local propagation requires that the causal graph be singly-connected [Pearl, 1988b]. But where applicable, it provides a rather simple scheme to determine the effect of observing a variable e on any other variable z in the network, in terms of increasing or decreasing the probability of Z (if a proposition), or more generally the sign of the influence $S^\delta(e, z)$. This problem is diagrammed schematically in Figure 6.

When the reasoning is purely predictive (following the arrows from tail to head) or diagnostic (in the reverse direction), or it traverses a divergent node (where two links join tail to tail), simple chaining will work. That is, sign multiplication along the path of qualitative influences between e and z yields the required sign of the influence $S^\delta(e, z)$. If there is no evidential support for a convergent node (where two links join head to head), then its neighboring causes are independent, and so the intercausal link contributes a zero, rendering adjoining paths independent. What this paper contributes is the means to traverse convergent nodes for which there is evidential support. If the conver-

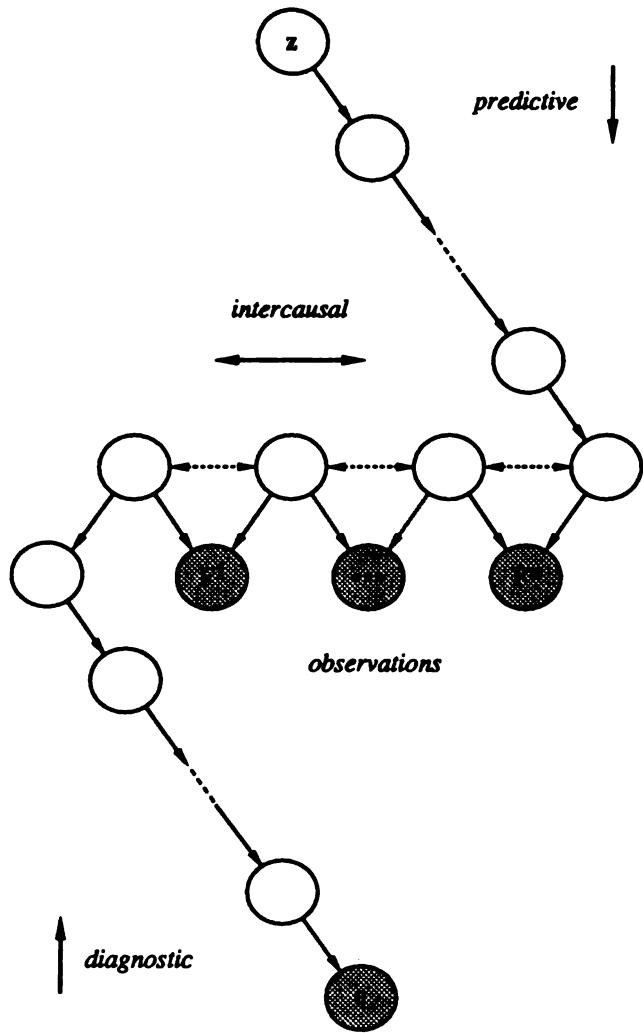


Figure 6: Propagating the impact of evidence e on some distal cause z given observations F^1, \dots, F^n requires a mixture of predictive, diagnostic, and intercausal reasoning.

gent node has been observed, the sign of the product synergy determines the sign of the intercausal link, which replaces the two converging links in the influence chaining. When there is partial evidential support for the convergent node, Theorem 6 provides the conditions under which the sign of the intercausal link can be determined. Assuming the signs of all intercausal links in the path between e and z are determinable, they are simply included in the product with all the other links in the path to determine its overall sign.

9 Occam's Razor and Intercausal Reasoning

As mentioned above, explaining away is closely related to Occam's razor and the principle of parsimony.

Suppose there are several causal hypotheses—each of which could explain an observed effect by itself—related to the finding according to the noisy OR relationship. The negative product synergy of the noisy OR gives rise to negative intercausal relations between each pair of hypotheses given the finding. Thus, invoking one hypothesis reduces belief in the others. This process is analogous to the action of Occam's razor in slicing away unnecessary hypotheses.

In addition, the analysis of intercausal reasoning presented here suggests that a somewhat richer notion of parsimony may be appropriate. If two or more causes interact with a positive product synergy in explaining a finding, thus producing a positive intercausal relation, their joint occurrence may be a better explanation of the finding than the occurrence of either alone. The synergistic effects of drinking and driving, and of cat allergies and cats are two examples. One might be tempted to invoke "Occam's glue" in such cases, as the multiple hypotheses adhere together to form a coherent scenario. But perhaps it is more appealing to regard the conjunctive relation as suggesting their combination as a single compound hypothesis. Seen in this light they are not being multiplied beyond necessity, and so not actually contravening the principle of parsimony.

Note that when there are multiple evidence variables, positive intercausal relationships and complementary hypotheses can arise even in situations consisting solely of noisy OR relations and negative synergy. Consider the QPN in Figure 7a, where three diseases, represented by propositional variables d^1, d^2 , and d^3 , can variously account for two findings, f^1 and f^2 . Suppose that all links are positive, and the two links to each finding combine in a manner satisfying negative product synergy (e.g., as a noisy OR). According to Theorem 1, given both findings f^1 and f^2 , we obtain the two negative intercausal influences $S^-(d^1, d^2), S^-(d^2, d^3)$, depicted in Figure 7b. Chaining these, we can conclude $S^+(d^1, d^3)$ upon removal of d^2 , indicating that events D^1 or D^3 are complementary. This conclusion fits the intuitive observation that the findings can be explained either by the single disease D^2 or by the combination D^1 and D^3 . If D^1 and D^3 are common diseases and D^2 relatively rare, it is quite possible that the combination is more probable than the single disease. Thus the intercausal analysis dictates how causal events should be clustered in compound hypotheses. Events complementary in the causal explanation become related by positive influences, without explicit set-covering computations.

10 Conclusions

Intercausal relations play a central role in the combination of diagnostic and predictive reasoning. The qualitatively significant property of interacting hypothe-

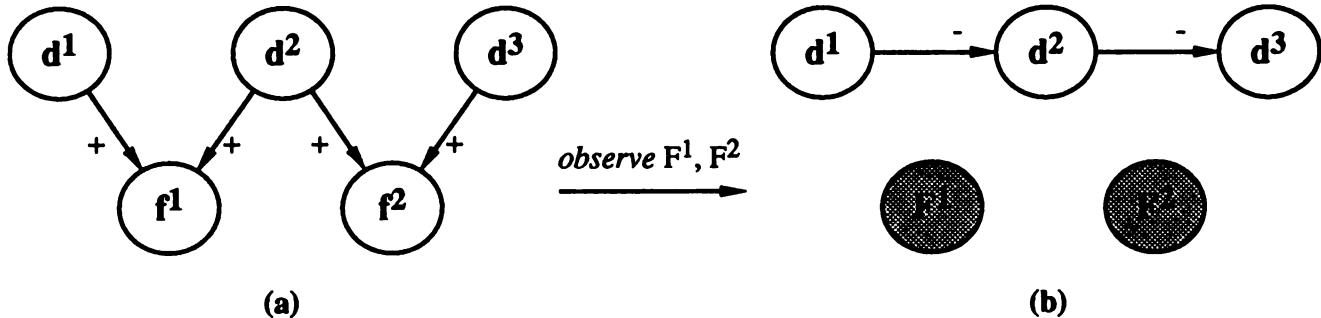


Figure 7: (a) A QPN with three diseases that can cause two findings. (b) Observing findings F^1 and F^2 produces two negative intercausal influences, which can be chained to reveal a positive relation between D^1 and D^2 .

ses is whether they compete with or complement each other in explaining the observed findings. In the former case, one cause “explains away” the other given the observation. However, we have shown via example and probabilistic analysis that explaining away is not the only pattern of intercausal reasoning. To account for this distinction, we have derived a general probabilistic criterion, *negative product synergy*, that precisely justifies explaining away. Further analysis reveals the relation of product synergy to other types of synergy and to canonical interaction models such as noisy OR. New inference rules for combining product synergy with other qualitative relations extend the QPN formalism to support qualitative intercausal reasoning. These rules may be incorporated both in reduction- and propagation-based inference algorithms to support a stronger set of qualitative inferences.

The main appeal of QPNs is that they require less precision than numerical probabilistic systems while capturing some of their most significant behaviors. But qualitative probabilistic inference may be useful even for numerical systems, as a way of simplifying the reasoning for explanation to human users in a way that may be more compatible with their own thought processes [Henrion and Drusdzel, 1990]. In that sense, the qualitative schema for intercausal relations serve as a descriptive model, accounting for an important pattern of commonsense reasoning not adequately captured by other approaches.

We also believe that it may be computationally advantageous to maintain these qualitative distinctions even when numeric information is available. As described in Section 9, intercausal relations qualitatively restrict the reasonable patterns in which to cluster events in compound hypotheses. These constraints can be exploited in diagnosis to prune the space of composite hypotheses at a high-level, based on qualitative admissibility. We intend to investigate this prospect in further work.

A Proofs

Theorem 1 Let a , b , and x be the predecessors of c such that a and b are conditionally independent given y , that is, $S^0(a, b)$. A necessary and sufficient condition for $S^-(a, b)$ upon observation of c_0 is negative product synergy, $X^-(\{a, b\}, c_0)$.

PROOF The distribution for a given b , x , and y upon observation of c_0 is, by Bayes’s rule,

$$f_a(a|bc_0xy) = \frac{f_c(c_0|abxy)f_a(a|bxy)}{f_c(c_0|bxy)}. \quad (7)$$

By conditional independence, we can drop the y condition from f_c terms, and the x condition from the f_a term on the right-hand side. The qualitative influence of a on b is positive iff (7) obeys the monotone likelihood ratio property (1), and negative iff the inequality of (1) is reversed. Substituting (7) in the likelihood ratio for a_i given a pair of values for b , $b_1 > b_2$, we get

$$\frac{f_c(c_0|a;b_1x)f_a(a_i|b_1y)f_c(c_0|b_2x)}{f_c(c_0|a;b_2x)f_a(a_i|b_2y)f_c(c_0|b_1x)}. \quad (8)$$

Since $f_c(c_0|b_jx)$ does not depend on a_i , the ratio (8) is increasing or decreasing in a_i in direct correspondence with

$$\frac{f_c(c_0|a;b_1x)f_a(a_i|b_1y)}{f_c(c_0|a;b_2x)f_a(a_i|b_2y)}. \quad (9)$$

By the conditional independence of a and b given y (the S^0 condition), $f_a(a_i|b_1y) = f_a(a_i|b_2y)$, so these terms may be cancelled from the expression, leaving

$$\frac{f_c(c_0|a;b_1x)}{f_c(c_0|a;b_2x)}.$$

The direction of change of this expression with respect to a_i is exactly the product synergy condition. \square

Theorem 2 Let a , b , and x be the predecessors of c . A sufficient condition for $S^b(a, b)$ upon observation of c_0 is an influence $S^b(a, b)$ prior to the observation plus a product synergy relation, $X^b(\{a, b\}, c_0)$, of compatible sign.

PROOF Proceed as for Theorem 1, up to the reference to unconditional independence. The ratio (9) can be factored into two parts,

$$\left(\frac{f_c(c_0|a_1 b_1 x)}{f_c(c_0|a_1 b_2 x)} \right) \left(\frac{f_a(a_1|b_1 y)}{f_a(a_1|b_2 y)} \right).$$

The first part increases according to the sign of product synergy, the second contingent on the direct influence of a on b prior to observation of c_0 . When the two agree, the direction of the entire expression is determined, establishing the qualitative influence of a on b posterior to the observation. \square

Theorems 3 and 4 are corollaries of the two somewhat more general theorems 7 and 8, respectively, presented and proved below. The conditions for Theorem 7 (Theorem 8) are automatically satisfied when the two influences have the same sign (opposite signs).

Theorem 7 If, for each $a_1 > a_2$, $b_1 > b_2$, and x , one of the following holds:

1. $G_{1,1} \geq G_{1,2}$ and $G_{1,1} \geq G_{2,1}$
2. $G_{1,1} \leq G_{1,2}$ and $G_{1,1} \leq G_{2,1}$
3. $G_{2,2} \geq G_{1,2}$ and $G_{2,2} \geq G_{2,1}$
4. $G_{2,2} \leq G_{1,2}$ and $G_{2,2} \leq G_{2,1}$

where $G_{i,j} = \Pr(c \geq c_0|a_i b_j x)$,

$$\text{Then } X^+(\{a, b\}, C) \Rightarrow Y^+(\{a, b\}, c) \text{ and } Y^-(\{a, b\}, c) \Rightarrow X^-(\{a, b\}, C).$$

PROOF ($X \Rightarrow Y$): Suppose positive product synergy holds. Therefore,

$$G_{1,1}G_{2,2} \geq G_{1,2}G_{2,1}. \quad (10)$$

We can use this inequality to bound the sum of $G_{1,1}$ and $G_{2,2}$,

$$G_{1,1} + G_{2,2} \geq G_{1,2} + (G_{1,1} - G_{1,2}) + \frac{G_{1,2}G_{2,1}}{G_{1,1}}. \quad (11)$$

Rearranging, the right-hand side of (11) can be rewritten as

$$G_{1,2} + G_{2,1} + \left[(G_{1,1} - G_{1,2}) - \frac{(G_{1,1} - G_{1,2})}{G_{1,2}} G_{2,1} \right]. \quad (12)$$

The bracketed expression of (12) is guaranteed non-negative under either of the first two conditions of the theorem, establishing the inequality required for additive synergy. Noting that the entire argument is symmetric in $G_{1,1}$ and $G_{2,2}$, we get the remaining conditions.

($Y \Rightarrow X$): By a similar analysis, negative additive synergy provides the following bound on the product of $G_{1,1}$ and $G_{2,2}$:

$$G_{1,1}G_{2,2} \leq G_{1,2}G_{1,1} + G_{2,1}G_{1,1} - G_{1,1}^2.$$

The right-hand side can be rewritten as

$$G_{1,2}G_{2,1} + [(G_{1,1} - G_{2,1})(G_{1,2} - G_{1,1})]. \quad (13)$$

The bracketed expression is guaranteed non-positive under either of the first two conditions, establishing the upper bound required for negative product synergy. A symmetric argument substituting $G_{2,2}$ for $G_{1,1}$ validates the latter two conditions. \square

Theorem 8 If, for each $a_1 > a_2$, $b_1 > b_2$, and x , one of the following holds:

1. $G_{1,1} \leq G_{1,2}$ and $G_{1,1} \geq G_{2,1}$
2. $G_{1,1} \geq G_{1,2}$ and $G_{1,1} \leq G_{2,1}$
3. $G_{2,2} \leq G_{1,2}$ and $G_{2,2} \geq G_{2,1}$
4. $G_{2,2} \geq G_{1,2}$ and $G_{2,2} \leq G_{2,1}$

$$\text{Then } X^-(\{a, b\}, C) \Rightarrow Y^-(\{a, b\}, c) \text{ and } Y^+(\{a, b\}, c) \Rightarrow X^+(\{a, b\}, C).$$

PROOF ($X \Rightarrow Y$): We proceed as for the previous theorem, achieving an upper bound for $G_{1,1} + G_{2,2}$ equivalent to (12). The first two conditions are sufficient for the bracketed expression to be negative, and the second two perform a corresponding role in the symmetric argument that switches $G_{1,1}$ and $G_{2,2}$.

($Y \Rightarrow X$): An analogous adaptation of the previous theorem, using expression (13), justifies this result. \square

Theorem 5

$$X^\delta(\{a, b\}, C) \wedge Y^{-\delta}(\{a, b\}, c) \Rightarrow X^\delta(\{a, b\}, \bar{C}).$$

PROOF Using the notation from above, the condition for $X^+(\{a, b\}, C)$ is given by (10). The corresponding condition for $X^+(\{a, b\}, \bar{C})$ is

$$(1 - G_{1,1})(1 - G_{2,2}) \geq (1 - G_{1,2})(1 - G_{2,1}),$$

or equivalently,

$$G_{1,1}G_{2,2} - G_{1,2}G_{2,1} \geq (G_{1,1} + G_{2,2}) - (G_{1,2} + G_{2,1}).$$

Positive product synergy (X^+) guarantees that the left-hand side is positive, and negative additive synergy (Y^-) that the right-hand side is negative, establishing the inequality. The case for X^- and Y^+ is analogous. \square

Theorem 6 Suppose $X^{\delta_1}(\{a, b\}, C)$, $Y^{\delta_2}(\{a, b\}, c)$, and $S^{\delta_3}(c, e)$ hold in a network with no direct links from a or b to e . Then $X^{\delta_1}(\{a, b\}, E)$ holds in the network after reducing c if either:

1. $\delta_1 = \delta_2$ and $\delta_3 = +$, or
2. $\delta_1 = -\delta_2$ and $\delta_3 = -$.

PROOF Let $H_{i,j} = \Pr(E|a_i b_j z)$ and, as before, $G_{i,j} = \Pr(C|a_i b_j z)$. Since e is conditionally independent of a and b given c ,

$$H_{i,j} = \Pr(E|C)G_{i,j} + \Pr(E|\bar{C})(1 - G_{i,j}).$$

Expanding terms and simplifying, the product of two H expressions is

$$H_{i,j} H_{k,l} = G_{i,j} G_{k,l} \Delta^2 + (G_{i,j} + G_{k,l}) \Pr(E|\bar{C})\Delta,$$

where $\Delta = [\Pr(E|C) - \Pr(E|\bar{C})]$, which is positive or negative according to δ_3 . Since Δ^2 is always positive, the comparison of a pair of H products is the same as for the corresponding G products if the comparison of second additive terms also agrees. When $\Delta > 0$, the sign of this second comparison is determined by the additive synergy relation, and when $\Delta < 0$, by its negation. \square

Acknowledgments

We thank Ramesh Patil and an anonymous reviewer for valuable suggestions. Max Henrion is supported by the National Science Foundation under grant IRI-8807061 to Carnegie Mellon and by the Rockwell International Science Center.

References

- [Geffner, 1988] Hector Geffner. On the logic of defaults. In *Proceedings of the National Conference on Artificial Intelligence*, pages 449–454, St. Paul, MN, 1988. AAAI.
- [Geffner, 1990] Hector Geffner. Causal theories for nonmonotonic reasoning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 524–530, Boston, MA, 1990. AAAI.
- [Henrion, 1987] Max Henrion. Uncertainty in Artificial Intelligence: Is probability epistemologically and heuristically adequate? In J. Mumpower et al., editors, *Expert Judgment and Expert Systems*, volume 35 of *NATO ISI Series F*, pages 105–130. Springer-Verlag, Berlin, 1987.
- [Henrion, 1989] Max Henrion. Some practical issues in constructing belief networks. In Laveen N. Kanal, Tod S. Levitt, and John F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*. North-Holland, 1989.
- [Henrion and Drusdzel, 1990] Max Henrion and Marek J. Drusdzel. Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 10–20, Cambridge, MA, 1990.
- [Karlin, 1968] Samuel Karlin. *Total Positivity*. Stanford University Press, 1968.
- [Karlin and Rinott, 1980] Samuel Karlin and Yosef Rinott. Classes of orderings of measures and related correlation inequalities. i. multivariate totally positive distributions. *Journal of Multivariate Analysis*, 10:467–498, 1980.
- [Michelena and Agogino, 1989] Nestor Michelena and Alice Agogino. Deterministic monotonic influence diagrams. Working Paper 89-1101-1, Berkeley Expert Systems Technology Laboratory, December 1989.
- [Milgrom, 1981] Paul R. Milgrom. Good news and bad news: Representation theorems and applications. *Bell Journal of Economics*, 12:380–391, 1981.
- [Pearl, 1988a] Judea Pearl. Embracing causality in default reasoning. *Artificial Intelligence*, 35:259–271, 1988.
- [Pearl, 1988b] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- [Pearl et al., 1989] Judea Pearl, Dan Geiger, and Thomas Verma. Conditional independence and its representations. *Kybernetika*, 25:33–44, 1989.
- [Shachter, 1989] Ross D. Shachter. Evidence absorption and propagation through evidence reversals. In *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*, pages 303–310, Windsor, ON, 1989.
- [Wellman, 1990a] Michael P. Wellman. *Formulation of Tradeoffs in Planning Under Uncertainty*. Pitman, London, 1990.
- [Wellman, 1990b] Michael P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44:257–303, 1990.
- [Wellman, 1990c] Michael P. Wellman. Graphical inference in qualitative probabilistic networks. *Networks*, 20:687–701, 1990.

Qualitative Simulation with Multivariate Constraints

Michael P. Wellman
USAF Wright Laboratory
WL/AAA-1
Wright-Patterson AFB, OH 45433
wellman@wrdc.af.mil

Abstract

Existing techniques for qualitative simulation (e.g., QSIM) provide a limited capability for expressing and reasoning about multivariate functions. In this paper, I develop a general scheme for monotonicity constraints among an arbitrary number of variables, and show how it can be incorporated within the QSIM framework. Applying the new constraint extends the scope of qualitative models by relaxing the additivity assumption, with no concomitant loss of inferential power. Although it is impossible to fully enforce interval-sensitive relations among variables that are only ordinally scaled, additivity constraints can justify further pruning when the space of known corresponding values is sufficiently dense. To support this behavior, I introduce higher-order qualitative constraints that distinguish among variable interactions that are synergistic, additive, or sub-synergistic. These qualitative synergy concepts are a straightforward extension of the basic monotonicity constraints, and capture a variety of common patterns in qualitative reasoning.

1 Qualitative Simulation

The task of *qualitative dynamical analysis* is to determine the qualitatively distinct potential behaviors of a dynamic system, which is itself specified in qualitative terms. The qualitative description of the system generally consists of constraints among a set of state variables and their time derivatives. Following Kuipers [1986], we call this constraint system a *qualitative differential equation* (QDE), and interpret it semantically as the class of systems of ordinary differential equations consistent with the constraints. The output of qualitative analysis (sometimes called an *envisagement*) describes a set of abstract behaviors, each

corresponding to a class of precise behaviors of the dynamic system. We say that a method for qualitative dynamic analysis is *sound* iff the behavior of every system consistent with the QDE is abstracted by one of the qualitative behaviors, and that the method is *complete* iff every abstract behavior is realized by one of these systems.¹

Abstract behaviors are often represented as sequences of *qualitative states*—classes of situations that partition the overall state space. Constraints dictate which state transitions are admissible, that is, which qualitative states may succeed a given state. In a *simulation* approach to dynamic analysis, the reasoner begins at the initial qualitative state and sweeps forward in time, applying transition analysis to determine the possible paths of qualitative states that the system may traverse. This approach contrasts with attempts to derive global properties of the system directly, for example by reasoning about stability or periodicity without necessarily enumerating the stable or periodic behaviors. The two approaches are complementary and may be combined, for example by applying the results of global analysis as a filter on the behaviors generated via simulation. For an elaboration of this framework, see Sacks [1990].

The power and computational difficulty of a particular analysis task depends critically on the expressiveness of the qualitative representation. The representation language dictates what types of constraints can be expressed in the QDE, and what distinctions are maintained in qualitative behaviors. “Qualitative” is a relative term, dependent on which “qualities” are consid-

¹These terms are sometimes reversed in the AI literature on qualitative reasoning [Weld and de Kleer, 1989]. Both terminologies are correct with respect to some interpretation of the qualitative dynamic analysis process as theorem proving. I adopt the stated convention because it corresponds more naturally with the inference by constraint filtering or pruning typical of qualitative simulation. A filter is considered sound if it prunes only inconsistent solutions (i.e., the constraints before pruning entail the constraints after), and complete if it prunes all of them.

ered important in a given context. The above task description applies to a broad spectrum of analysis problems, ranging from the most precise, numeric representations (degenerate qualitative analysis) to the extremely abstract representations investigated by AI researchers.

In this paper, I examine some fundamental representation issues at the abstract end of this qualitative analysis spectrum. In particular, I develop a general mechanism for monotonicity constraints on multivariate functions based on partial derivatives. For this purpose I adopt the conceptual machinery of Kuipers's QSIM [1986], a general framework for qualitative simulation that is mathematically well-characterized and widely used by researchers experimenting with qualitative analysis. The extended scheme permits modelers to avoid superfluous additivity assumptions, resulting in QDEs more accurately matched to their qualitative behaviors. Straightforward extensions permit expression of higher-order interactions among variables, introducing a rich set of qualitative relations heretofore unexploited in qualitative reasoning.

2 Monotonicity Constraints

One of the basic types of constraint employed in qualitative reasoning is *monotonicity*. Monotonicity constraints restrict the direction of interaction among state variables without specifying anything further about the relation's shape or functional form. Variables constrained solely by monotonicity conditions are effectively only *ordinally scaled*; relations involving the variables are invariant with respect to order-preserving transformations [Krantz et al., 1971]. This property meshes nicely with *quantity space* representations of variable domains, where qualitative states are defined by the ordinal relation of variables to designated landmark values.

The appeal of monotonicity constraints is that information about the direction of interaction is often more available and less context-sensitive than specifications requiring higher precision. Similarly, ordinal relations are easier to verify and more robust than relations dependent on magnitudes and differences. Because their premises are strictly weaker, conclusions derived solely from monotonicity can be taken as valid in a broader class of circumstances than those based on more precise models.

For these and other reasons, most AI work on qualitative reasoning has featured ordinal quantity spaces with monotonicity constraints on functional relations. For example, the binary M^+ constraint employed by QSIM [Kuipers, 1986] dictates that one variable be a strictly increasing function of the other, at all points in time. Let x, y, \dots be state variables, and $x(t), y(t), \dots$

denote their values at some time t .²

Definition 1 (M^+ Kuipers [1986]) $M^+(x, y)$ iff there exists a differentiable function f such that for all t , $y(t) = f(x(t))$ and $f'(x) > 0$.

Because under the stated conditions f is invertible, M^+ is symmetric in x and y . The constraint M^- is defined similarly, with $f'(x) < 0$. I use the notation M^δ to denote the lack of a constraint on f 's derivative, and M^δ to parameterize the monotonicity relation, with $\delta \in \{+, -, ?\}$.

QSIM enforces constraints by filtering qualitative states and state transitions. Each variable is represented in a state by its qualitative magnitude (position with respect to an ordered set of landmark values) and its direction of change (the sign of its time derivative, called a *qdir*, one of *inc*, *std*, or *dec*). To satisfy the constraint $M^+(x, y)$, variables y and x must have the same *qdir*, and their qualitative magnitudes must be consistent with known points on the function f . These points, called *corresponding values*, are (x, y) pairs given as part of the QDE or discovered during qualitative simulation. If (a, b) are corresponding values for (x, y) , then $b = f(a)$, and both values belong to the landmark set of their respective variables. The state transition filter for M^δ ensures that y and x reach these corresponding landmarks simultaneously, or not at all. This correspondence requirement follows from the one-to-one functional relation of y and x .

3 Multivariate Functions

Forbus's Qualitative Process Theory (QPT) [Forbus, 1984] makes use of another type of monotonic relation called *qualitative proportionality*, or Q^δ .³ The constraint $Q^\delta(x, y)$ dictates a monotonic relation between y and x , all else being equal. Unlike M^δ , Q^δ allows that y may depend on other variables in addition to x , and hence there might not exist a function f on x satisfying Definition 1 above. Instead, Q^+ constrains the *partial* derivative of a *multivariate* function g , that is, $y(t) = g(\dots, x(t), \dots)$ and $\partial g / \partial x > 0$ at all points in g 's domain.

To account for these partially monotonic relations in qualitative simulation, we need a method for encoding

²Kuipers calls these variables *parameters*, and requires that they be *reasonable* functions of time, which means that they are continuously differentiable and have a finite number of critical points. In this discussion we also assume reasonableness. See [Kuipers, 1986, Definition 2.1] for the details.

³As above, δ ranges over $\{+, -, ?\}$. I follow Crawford et al. [1990] in substituting Q^δ for Forbus's notation α_{Q^δ} to maintain typographic uniformity. QPT's "direct influence" relation I^δ presents issues similar to Q^δ , and is not discussed here.

them as constraints in a QDE representation. Crawford et al. [1990] consider this problem in their work on QPC, a compiler from QPT models to QSIM QDEs. There is no direct mapping because QSIM does not support partial derivative constraints, and can express only those multivariate functions composed of sums and products.⁴ The difficulty with partial derivative constraints is that they involve a set of “other variables” implicitly held constant, but these other variables are not identified in isolated Q^δ assertions.

Forbus’s QPE [1989] operates over QPT models by collecting the Q^δ constraints active at a given time and applying a closed-world assumption to infer the signs of total derivatives for variables of interest, where possible. QPC employs a similar technique to identify the active relations, but rather than attempt to resolve them immediately, translates the relations into QSIM constraints and assembles them in a QDE. Given $Q^{\delta_i}(x_i, y)$, $i = 1, \dots, n$, QPC creates n constraints $M^{\delta_i}(x_i, y_i)$, introducing intermediate variables y_i and implicit functions f_i over the x_i , each monotonic in the direction indicated by δ_i . Next, the y_i are related to y by QSIM’s ADD constraint,⁵ so that

$$y(t) = \sum_{i=1}^n y_i(t) = \sum_{i=1}^n f_i(x_i(t)).$$

This translation enforces the partial derivative constraints; that is,

$$\frac{\partial y}{\partial x_i} = \frac{\partial y_i}{\partial x_i} = f'_i(x_i),$$

which is positive or negative in accordance with δ_i . However, it is also more restrictive than the original monotonicity constraints, in that the latter do not require that y be separable into a linear combination of univariate functions of the x_i .⁶

The inability to express partial monotonicity constraints on arbitrary multivariate functions is a fundamental limitation of the QSIM QDE representation. Fortunately, it appears that this omission can be remedied by a straightforward generalization of the existing

⁴It has come to my attention (Kuipers, personal communication) that the QSIM implementation has recently been extended to incorporate multivariate monotonicity constraints as described in this paper (using equivalent filtering rules, developed independently). References to QSIM and QPC herein should be understood as the earlier versions of those methods, described in the papers cited.

⁵In QSIM, ADD is defined as a ternary constraint: $ADD(z, y, z)$ means that $z(t) + y(t) = z(t)$. Sums of an arbitrary number of terms can be created by nesting ADD constraints in a hierarchical structure.

⁶Crawford et al. [1990] claim that this additivity assumption is a part of QPT. Reading Forbus [1984; 1989], it appears that QPT does assume additivity of direct influences (I^δ), but not indirect ones (Q^δ). In any case, the restriction does not seem sufficiently motivated, and is often unnecessary, as demonstrated below.

monotonicity relation. In the next section, I present such a generalisation, and show that it can expand the scope of a given qualitative analysis without degrading the strength of its conclusions.

4 Partial Monotonicity

Let $\Delta = (\delta_1, \dots, \delta_n)$ be a vector of qualitative signs. Let $>_\delta$ denote the relation $>$, $<$, $=$, or any (the vacuous relation holding for all values) as δ is $+$, $-$, 0 , or $?$. Then we can define the multivariate monotonicity constraint as follows.

Definition 2 (M^Δ) $M^\Delta(x_1, \dots, x_n, y)$ iff there exists a continuous and differentiable function g such that for all t , $y(t) = g(x_1(t), \dots, x_n(t))$ and

$$\frac{\partial g}{\partial x_i} >_\delta 0, \quad i = 1, \dots, n,$$

at all points in the domain of g .

We need not consider the qualitative sign $\delta = 0$, because if $\partial g / \partial x$ is uniformly zero, then g does not really depend on x .

The vector Δ can be interpreted as a qualitative abstraction of the gradient $\nabla g = (\partial g / \partial x_1, \dots, \partial g / \partial x_n)$. An $M^\Delta(x_1, \dots, x_n, y)$ constraint establishes a functional relationship between y and the x_i , and dictates that the dependence is monotone according to the qualitative gradient.⁷ For example, such a constraint with $\Delta = (+, +)$ asserts that y is a bivariate function of x_1 and x_2 , increasing in both arguments. Note that $ADD(x_1, x_2, y)$ implies $M^{(+,+)}(x_1, x_2, y)$, but the converse is not valid. The monotonicity condition is strictly more general than additive decomposition.

M^Δ directly constrains the legal combinations of qdirs for y and the x_i . We can express the total derivative of y with respect to time in terms of partial derivatives on the x_i :

$$\frac{dy}{dt} = \sum_{i=1}^n \frac{\partial y}{\partial x_i} \frac{dx_i}{dt}. \quad (1)$$

The signs of the derivatives dx_i/dt are determined by $qdir(x_i)$, whereas the δ_i constrain the signs of $\partial y / \partial x_i$. If all of the summed terms on the right-hand side of (1) agree in sign (zero terms may be ignored), then dy/dt must agree as well. The following proposition⁸ provides a complete statement of the direction-of-change compatibility requirements imposed by the M^Δ constraint.

⁷Dangelmaier [1989] presents methods for reasoning about the qualitative gradient and other multivariate derivatives as state variables in a qualitative analysis. Such an approach appears to be complementary to the one presented here.

⁸Proofs of all results are provided in the appendix.

Proposition 1 A qualitative state is qdir-consistent with respect to $M^\Delta(x_1, \dots, x_n, y)$ iff the following conditions are satisfied:

1. $\forall i [qdir(x_i) = std] \rightarrow qdir(y) = std.$
2. $\forall i [qdir(x_i) = std \vee (qdir(x_i) = inc \wedge \delta_i = +) \vee (qdir(x_i) = dec \wedge \delta_i = -)] \wedge \exists j [qdir(x_j) \neq std] \rightarrow qdir(y) = inc.$
3. $\forall i [qdir(x_i) = std \vee (qdir(x_i) = dec \wedge \delta_i = +) \vee (qdir(x_i) = inc \wedge \delta_i = -)] \wedge \exists j [qdir(x_j) \neq std] \rightarrow qdir(y) = dec.$

Notice that none of the conditions' antecedents apply when two or more of the expressions on the right-hand side of (1) disagree in sign. This can happen, for example, when two variables have the same qdir but opposite δ s, or when $\delta_i = ?$ and $qdir(x_i) \neq std$. In such cases, the sign of dy/dt is ambiguous, and magnitude information would be required to resolve $qdir(y)$.

The conditions of Proposition 1 also explain why we need to encode partial monotonicity as a multivariate constraint, rather than a collection of bivariate constraints. A single δ_i says nothing about the compatibility of $qdir(x_i)$ and $qdir(y)$, unless the qdirs for all x_j , $j \neq i$, are *std*. For example, suppose we have $qdir(x_i) = inc$, $i = 1, \dots, n$, and n positivity constraints on the partial derivatives $\partial y / \partial x_i$. A qdir of *dec* for y is compatible with each of the $\delta_i = +$ constraints in isolation, and even with any proper subset of them. It is only the conjunction of all n that presents a conflict and allows us to prune the state as infeasible.

The consistency constraints on qualitative magnitude are directly analogous to the restrictions on direction-of-change. We replace the conditions on qdirs with predicates testing the ordinal relation of the variables to known corresponding values.

Definition 3 (cvt) A corresponding value tuple (cvt) for a constraint instance C is an assignment (w_1, \dots, w_m) to C 's variables z_1, \dots, z_m such that

1. (w_1, \dots, w_m) satisfies the function underlying that constraint, and
2. w_i is in the landmark set for variable z_i .

For example, a cvt for $M^\Delta(x_1, \dots, x_n, y)$ is a tuple (a_1, \dots, a_n, b) such that $b = g(a_1, \dots, a_n)$, for g the monotonic function underlying this M^Δ (Definition 2).

Proposition 2 Suppose (a_1, \dots, a_n, b) is a cvt for variables (x_1, \dots, x_n, y) . An assignment to the variables is consistent with respect to the cvt and constraint $M^\Delta(x_1, \dots, x_n, y)$ iff there is some i such that $\delta_i = ?$, or the following conditions are satisfied:

1. $\forall i [x_i = a_i] \rightarrow y = b.$
2. $\forall i [x_i \geq_{\delta_i} a_i] \wedge \exists j [x_j \neq a_j] \rightarrow y > b.$
3. $\forall i [a_i \geq_{\delta_i} x_i] \wedge \exists j [x_j \neq a_j] \rightarrow y < b.$

We can verify that the conditions of Proposition 2 hold in a given qualitative state by direct inspection of qualitative magnitudes. Since the corresponding values (a_1, \dots, a_n, b) are part of the landmark sets for their respective variables, the state determines each inequality appearing in the conditions. Although the proposition checks for magnitude consistency with each cvt individually, satisfying the conditions separately implies consistency with M^Δ and the collection of cvts taken as a whole.

Proposition 3 Let CV be a set of cvts consistent with a constraint $M^\Delta(x_1, \dots, x_n, y)$. Suppose a qualitative state satisfies the conditions of Proposition 2 for this constraint and each cvt in CV . Then for any point covered by this qualitative state, there exists a continuous function g that obeys the M^Δ constraint, satisfies all cvts in CV , and includes this point.

5 Incorporating M^Δ in QSIM

Augmenting QSIM's repertoire of constraints with M^Δ apparently expands the expressive power of QDEs by permitting multivariate functions with an arbitrary combination of monotonicity relations. The qualitative direction and magnitude propositions above can be implemented as filters on qualitative states, together enforcing the M^Δ constraint within the QSIM framework. These filters, along with the general continuity rules on QSIM transitions, prune all state transitions that are incompatible with a given M^Δ constraint and known corresponding values.

As noted above, the existing *ADD* constraint is a special case of $M^{(+,+)}$. The multivariate monotonicity constraint is strictly weaker, and as one would expect, QSIM's constraint filter for *ADD* prunes all the state transitions disallowed by Propositions 1 and 2 for $\Delta = (+, +)$. Somewhat surprisingly, however, the *ADD* filter is no stronger than this $M^{(+,+)}$ filter.

Proposition 4 The QSIM constraint $ADD(x_1, x_2, y)$ as described by Kuipers [1986] admits the same qualitative state transitions as $M^{(+,+)}(x_1, x_2, y)$.⁹

The fact that the stronger constraint yields no additional pruning of state transitions is a consequence of the qualitative characterization of behaviors. Indeed, the conditions of Propositions 1 and 2 are the

⁹This result is confirmed by experience with the QSIM implementation, in which the correspondence between the *ADD* filter and Proposition 2 is considerably more transparent.

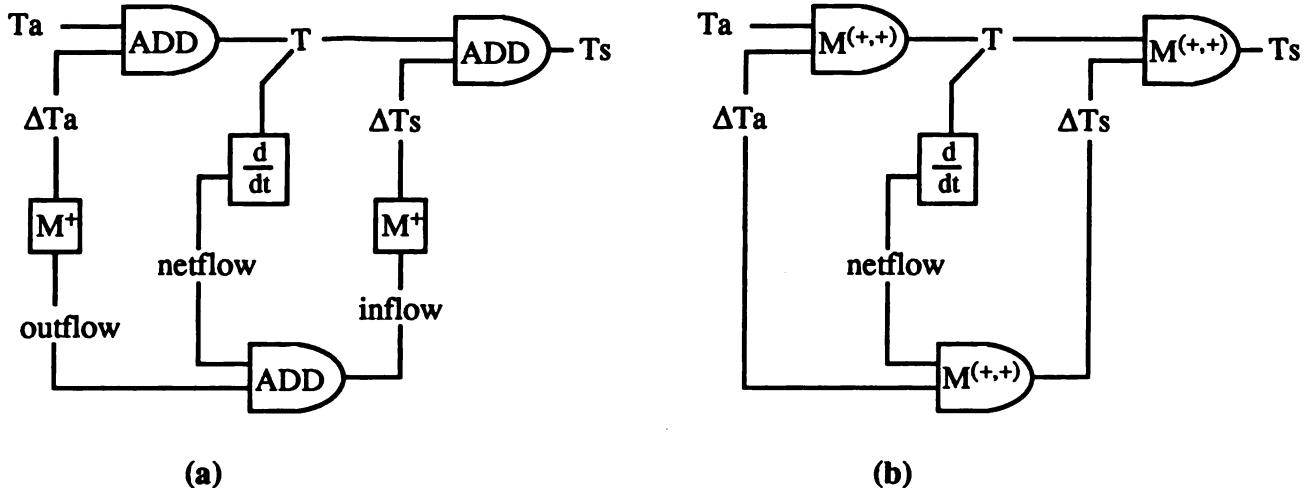


Figure 1: The double heat flow model: (a) original qsim version [Kuipers, 1984], and (b) revised model with ADD replaced by $M^{(+,+)}$.

strongest sanctioned by ADD for consistency of qdirs and qualitative magnitudes with an individual *cvt*.¹⁰ The fundamental reason behind the effective equivalence of ADD and $M^{(+,+)}$ in qsim is the ordinality of the variable scales. Quantities defined solely by their position in a sequence of landmarks are insensitive to monotone transformations, and hence we cannot fully enforce constraints that depend on difference relationships. In other words, ordinal variables have sufficient freedom to fit *any* functional form respecting their monotonicity constraints, including, of course, summation.

In rationalizing their use of ADD in translating QPT Q^δ relations, Crawford et al. [1990] observe that the linear decomposition assumption “does not cause problems in purely qualitative simulation.” Proposition 4 explains and justifies this claim, by demonstrating that qsim’s ADD really does not enforce additivity. A modified QPC employing M^Δ in place of the ADD/M^δ combination achieves the same results without appeal to the dubious postulate.

One immediate consequence of the proposition is that we can transform any qsim QDE by replacing all occurrences of ADD with an $M^{(++)}$ constraint, without affecting the result. This translation maintains soundness while extending the class of ODEs covered by the model, thereby enhancing the coverage of a given qualitative simulation. In other words, there are some behaviors generable by qsim that would be considered spurious with respect to ADD , but not with respect to $M^{(++)}$. An abstract example is provided at the end

of Section 6.

Furthermore, we may decrease the number of variables and constraints by collapsing other constraints (e.g., M^δ , $MINUS$, or nested ADD s) adjacent to the original ADD . For example, $M^-(z_1, z)$ and $ADD(z, z_2, y)$ can be replaced by the single constraint $M^{(-,+)}(z_1, z_2, y)$, eliminating variable z (assuming it participates in no other constraints, and is not of particular interest). This consolidation is similar to the simplification operations described by Kuipers [1984, Appendix D], although no effective (constraining) information is lost, except regarding the eliminated variables. In many cases, however, these variables are artificial, introduced by modelers to encode multivariate monotonicity relations in terms of the M^δ and ADD primitives provided by qsim. (Recall the intermediate variables y_i created by QPC in mapping Q^+ relations to the QDE representation.)

For example, consider the double heat flow model presented by Kuipers [1984]. In this system, there is a flow of heat from the source (at constant temperature T_s) to the object of interest (temperature T), and another flow from this object to the surrounding air (temperature T_a). Figure 1a depicts the original qsim model, and Figure 1b the model obtained by replacing ADD constraints with the weaker multivariate monotonicity relation, $M^{(+,+)}$. Corresponding values are retained. Notice that the translated version contains two fewer constraints and eliminates a pair of intermediate variables. The variables, *inflow* and *outflow*, are mere accounting quantities introduced to satisfy the ADD relation, of no intrinsic interest. The increased generality and economy of constructs in the revised model is achieved without loss of inferential power, as qsim produces the same set of potential behaviors under each

¹⁰However, the information that variables combine linearly does provide additional constraint on collections of *cvt*s. That is, Proposition 3 does not hold for ADD . This issue is explored further in Section 6.

of the qualitatively distinct initial conditions.

Noting that $\text{netflow} = \text{inflow} - \text{outflow}$, we could have translated the lowermost *ADD* constraint in Figure 1a to $M^{(+,-)}(\text{inflow}, \text{outflow}, \text{netflow})$, and, consolidating, $M^{(+,-)}(\Delta Ts, \Delta Ta, \text{netflow})$. In fact, this is a valid transformation regardless of additivity. $M^{(++,)}(x, y, z)$ is mathematically equivalent to $M^{(+,-)}(z, y, x)$, and indeed their respective constraint filters are identical. Similar updates of Δ can be derived for arbitrary permutations of the constraint's arguments. One ordering might be preferred over another on "causal" grounds, but there is no computational difference.¹¹

QSIM's *MULT* constraint is slightly more complicated than *ADD*, as the signs of partial derivatives depend on the absolute signs of the constraint's arguments. This can be handled by dividing the quantity space into its monotone regions. In many cases, the terms appearing in *MULT* constraints are naturally restricted to positive domains.

Another implication of multivariate constraints relates to the automatic generation of landmark values. In qualitative simulation, it is often useful to define new corresponding value tuples whenever one of the constrained variables reaches a critical value. However, the functions underlying multivariate constraints are generally not invertible, so there will be many value combinations for x_1, \dots, x_n corresponding to a particular y . Moreover, when the constraint may be nonlinear (as in M^Δ), the boundary in x_1, \dots, x_n separating qualitatively significant values of y is not a hyperplane but a *hypersurface* [Kokar, 1987], whose shape is restricted only by monotonicity. Reasoning about transitions across hypersurfaces is more complicated than for rectangular or polygonal regions, and may call for special techniques [Sacks, 1990].

6 Second-Order Partial Derivatives

The multivariate constraint $M^\Delta(x_1, \dots, x_n, y)$ enables us to assert the ordinal relationship between the output y of a function and each of its individual arguments, x_i . One natural extension of this facility would be some qualitative expression of the interaction among the x_i in their relation to y . For example, we may wish to declare that the x_i represent independent causes, enabling or inhibiting conditions, or synergistic factors. To do so, we require concepts characterizing the effect of one variable x_i on the relation between some others, for instance, x_j and y .

A local measure of this type of interaction is the

¹¹For a discussion of transformation rules on partial monotonicity conditions, albeit within a different scheme for constraint networks, see Michelena and Agogino [1989].

second-order mixed partial derivative,

$$\frac{\partial^2 y}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i} \left(\frac{\partial y}{\partial x_j} \right).$$

Just as for the first-order case, the sign abstraction of this derivative represents a qualitatively significant concept. Let y be a bivariate function of x_1 and x_2 . When the second-order mixed partial derivative is zero, the two variables combine independently, or additively.

$$y = g(x_1, x_2) = g_1(x_1) + g_2(x_2)$$

Positive or negative mixed partials indicate that the combination is more or less than additive. Let x'_i and x''_i be values of x_i such that $x'_i > x''_i$. Then a positive mixed partial derivative entails that

$$g(x'_1, x'_2) + g(x''_1, x''_2) > g(x'_1, x''_2) + g(x''_1, x'_2),$$

or in its equivalent difference form,

$$g(x'_1, x'_2) - g(x'_1, x''_2) > g(x''_1, x'_2) - g(x''_1, x''_2). \quad (2)$$

Functions satisfying (2) are called *supermodular* [Ross, 1983].¹² Reversing the inequality defines *submodular* functions, and making it an equality renders the function additive, or *modular*. We can extend this definition to functions with an arbitrary number of arguments by varying only two at a time in (2).

Examples abound where super- or sub-modularity—the direction of *synergy*—is a qualitatively significant distinction. An economic process involving a production function g should demand varying mixtures of inputs x_1 and x_2 depending on whether they are complementary or substitutable ingredients. Causal explanations often refer to enabling or inhibiting factors, which enhance or attenuate the influence of the causal variable on its effect.¹³ These situations can be distinguished by the mixed partial derivative of the effect with respect to the primary variable and its enabler/inhibitor. Note that the label "primary" is an artifact of the causal interpretation; the supermodularity relation itself is symmetric in x_1 and x_2 .

But perhaps the most promising potential application of the second-order relation is to the task of *qualitative control*, a prime motivation for performing qualitative analysis in the first place. When the objective is a supermodular function of a control variable and an observable, the *monotone decision property* dictates that the optimal control is an increasing function of the observable. While this constraint does not fix the

¹²Variants on the definition may differ on whether strictness is required in the inequality. The developments below may be adapted for either convention.

¹³For an analysis of probabilistic synergy concepts and their implications for abductive explanation, see [Wellman and Henrion, 1991].

control policy precisely, it prunes the space of qualitative strategies, just as qualitative dynamical analysis prunes qualitative behaviors.

To incorporate qualitative synergy into QSIM, we introduce a new constraint restricting the signs of second-order partial derivatives. Let Δ (note boldface) be an $n \times n$ matrix of δ_{ij} s. We define the constraint M^Δ by direct analogy to M^0 .

Definition 4 (M^Δ) $M^\Delta(x_1, \dots, x_n, y)$, $\Delta = (\delta_{ij})$, iff there exists a continuous and differentiable function g such that for all t , $y(t) = g(x_1(t), \dots, x_n(t))$ and

$$\frac{\partial^2 g}{\partial x_i \partial x_j} >_{\delta_{ij}} 0, \quad i, j = 1, \dots, n,$$

at all points in the domain of g .

Δ is the sign abstraction of the Hessian matrix, the two-dimensional counterpart of the gradient. We actually need specify only the upper or lower triangle of the matrix, as the second-order partial derivatives are symmetric for continuous functions.

The composition of Δ strongly constrains the qualitative form of g . For example, linear decomposition (i.e., $y = \sum g_i(x_i)$) follows from the signs $\delta_{ij} = 0$ for $i \neq j$. The diagonal elements of Δ , δ_{ii} , hinge on the concavity (-), convexity (+), or linearity (0) of g_i . By specifying both Δ and Δ , we can encode a generalized ADD constraint. $M^0(x_1, \dots, x_n, y)$ implies

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_n x_n,$$

and $M^{(\delta_1, \dots, \delta_n)}$ adds the further restrictions $\alpha_i >_{\delta_i} 0$, $i = 1, \dots, n$.

M^Δ can constrain ordinal relations among variable magnitudes if the space of known corresponding values is sufficiently dense. For example, if $\delta_{12} = 0$ then the difference in y associated with a particular change of x_2 is a constant independent of the fixed value of x_1 . Let a_1^1, a_1^2, \dots denote particular values of variable x_1 . Then $M^0(x_1, x_2, y)$ implies, among other things,

$$\begin{aligned} g(a_1^1, a_2^1) - g(a_1^1, a_2^2) &= g(a_1^3, a_2^1) - g(a_1^3, a_2^2), \text{ and} \\ g(a_1^1, a_2^1) - g(a_1^1, a_2^3) &= g(a_1^2, a_2^1) - g(a_1^2, a_2^3). \end{aligned}$$

Combining, we have

$$\begin{aligned} g(a_1^2, a_2^1) + g(a_1^1, a_2^3) + g(a_1^3, a_2^2) &= \\ g(a_1^1, a_2^2) + g(a_1^3, a_2^1) + g(a_1^2, a_2^3). \end{aligned} \quad (3)$$

Because the landmark values are totally ordered, cuts for this constraint define inequalities on instances of g . Suppose we know from corresponding values that

$$g(a_1^1, a_2^2) > g(a_1^2, a_2^1) \text{ and } g(a_1^3, a_2^1) > g(a_1^1, a_2^3).$$

Then (3) implies $g(a_1^3, a_2^2) > g(a_1^2, a_2^3)$. Thus, the collection of cuts in conjunction with the M^0 (generalized ADD) constraint justifies the pruning of any state

violating this condition. Similar arguments based on inequalities can be constructed for nonzero δ_{ij} s. For instance, the same conclusion would be valid in the example above if $\delta_{12} = +$ and $a_1^1 > a_1^2 > a_1^3$ for $i = 1, 2$.

We can also employ this example to demonstrate the potential spuriousness arising from interpreting $M^{(+,+)}$ as ADD in a QDE. Consider a QDE consisting of a single constraint, $ADD(x_1, x_2, y)$. Suppose the initial landmark sequences for x_1 , x_2 , and y are (in descending order)

$$\begin{aligned} x_1 &: (\infty, a_1^1, a_1^2, a_1^3, -\infty), \\ x_2 &: (\infty, a_2^1, a_2^2, a_2^3, -\infty), \\ y &: (\infty, b^{1,2}, b^{2,1}, b^{3,1}, b^{1,3}, b^{3,2}, -\infty). \end{aligned}$$

Suppose further that we are given corresponding values $(a_1^i, a_2^j, b^{i,j})$ for each $b^{i,j}$ in y 's landmark set. Finally, we take the initial state to have the variables at their corresponding values $(a_1^3, a_2^2, b^{3,2})$, with $qdir(x_1) = inc$, $qdir(x_2) = dec$, and any $qdir$ for y . When simulated, this QDE produces many possible behaviors, including some passing through a state where x_1 and x_2 reach their next landmarks (a_1^2 and a_2^3 , respectively), and y has increased to a point between its landmarks $b^{3,2}$ and $b^{1,3}$. However, we know from the analysis above that $a_1^3 + a_2^2 > a_1^2 + a_2^3$, hence this state is inconsistent with the corresponding values and $ADD(x_1, x_2, y)$. Yet such behaviors are not spurious with respect to the constraint $M^{(+,+)}(x_1, x_2, y)$, which suggests that the QSIM algorithm is strictly more powerful for QDEs defined by M^Δ in place of ADD.

In the next section, I describe how additivity and other second-order relations can be exploited by an extension of the qualitative simulation algorithm, without requiring numeric information.

7 Enforcing Second-Order Constraints

We can implement second-order relations in QSIM using the same filtering approach employed for other types of constraints. The general filter for M^Δ enforces the full set of difference relations induced by the constraint and known corresponding values. Let CV be a set of m cuts of the form (a_1, \dots, a_n, b) . Let $B = \{b^1, \dots, b^{|B|}\}$ be the set of values of y appearing in some tuple of CV , with $|B| \leq m$. Note that B is a subset of y 's landmark set, which imposes a total ordering on the b^i . Without loss of generality, suppose that the values are numbered so that $b^1 > \dots > b^{|B|}$. We can define value sets A_i and orderings $a_i^1 > \dots > a_i^{|A_i|}$ in a similar manner for each variable x_i .

Let the expression $g_{ij|cv}(x_i, x_j)$ denote the value of g as a function of x_i and x_j , with all other variables fixed at their level in the cut $cv = (a_1, \dots, a_n, b)$. That is,

$$g_{ij|cv}(x_i, x_j) \equiv g(x_1, \dots, x_n | x_k = a_k, k \neq i, j).$$

With each $cv \in CV$, we associate the following inequality schema, a pattern of instantiation for the supermodularity condition (2):

$$\begin{aligned} g_{ij|cv}(a_i^k, a_j^l) - g_{ij|cv}(a_i^k, a_j^{l+1}) &> \epsilon_{ij} \\ g_{ij|cv}(a_i^{k+1}, a_j^l) - g_{ij|cv}(a_i^{k+1}, a_j^{l+1}). \end{aligned} \quad (4)$$

The M^Δ constraint requires that (4) hold for $1 \leq i < j \leq n$, $1 \leq k \leq |A_i| - 1$, and $1 \leq l \leq |A_j| - 1$.

We can test a potential qualitative state, s , for consistency with M^Δ and CV by treating s as if it represented a *cvt*. Since the parameters in a qualitative state are completely ordered with respect to the existing landmarks, we can temporarily expand the sets B, A_1, \dots, A_n , if necessary, to include the corresponding values represented by the qualitative magnitude of s . Let $cvt(s)$ denote the provisional corresponding value tuple for qualitative state s .

Proposition 5 *Let B, A_1, \dots, A_n be decreasing sequences of values for variables y, z_1, \dots, z_n , as defined above, for the cvt collection $CV' = CV \cup \{cvt(s)\}$. State s is inconsistent with CV and the constraint $M^\Delta(x_1, \dots, x_n, y)$ if there is no ordered assignment to $b^1 > \dots > b^{|B|}$ satisfying the inequalities formed by instantiating the schema (4) for all $cv \in CV'$, for $1 \leq i < j \leq n$, $1 \leq k \leq |A_i| - 1$, and $1 \leq l \leq |A_j| - 1$.¹⁴*

Since the inequalities involved include only expressions of a single variable or difference between variables, the condition of Proposition 5 can be tested by solving a linear programming (LP) problem. To encode the problem in this form, we must identify each $g_{ij|cv}$ term in (4) with an LP variable, being careful to unify terms that are equivalent (e.g., those equal to some $b \in B$ according to CV). We can translate strict inequalities to nonstrict ones using an ϵ parameter representing the minimal increment that could make a difference for satisfiability (see Papadimitriou and Steiglitz for the derivation of valid bounds for ϵ [1982, Lemma 8.7]).

Because there is no objective function, we actually need only solve the “linear inequality” problem [Papadimitriou and Steiglitz, 1982], which may be easier than the general LP. Moreover, the special structure of the inequalities required for M^Δ filtering—LP variables appear in inequalities only with coefficients ± 1 , and the sole nonzero constant term is ϵ —may justify further simplifications. In any case, the problem is polynomial in the number of LP variables and inequalities, which in turn is polynomial in the number of *cvt*s and qualitative state variables. Specifically, the LP

¹⁴Although I have no proof that this filter is complete with respect to CV , I conjecture that it is, and note that the set of inequalities defined by the proposition entail (by transitivity) all other inequalities on y 's landmark values induced by the M^Δ constraint.

formulation of Proposition 5 requires $O(n^2m^3)$ linear inequalities among a quantity of variables of the same order.

Note that the constraint filter for M^Δ makes no use of Δ 's diagonal elements, δ_{ii} . Indeed, convexity assertions can provide no constraint on the qualitative magnitudes of strictly ordinal variables. However, the δ_{ii} can support state transition filtering for qualitative simulation when higher-order time derivatives are taken into account [Kuipers et al., 1989]. Let $qdir2(x)$ denote the direction of change (*inc*, *std*, *dec*) of x 's first derivative with respect to time.

Proposition 6 *Given a bivariate monotonicity constraint $M^\delta(x, y)$ and an associated second-order constraint $M^\Delta(x, y)$, where Δ has only a single element, δ^2 ,*

$$qdir2(y) = \delta^2 \oplus (\delta \otimes qdir2(x)), \quad (5)$$

where \oplus and \otimes denote sign addition and multiplication, respectively.

The essence of this result is contained in Proposition 5 of Kuipers et al. [1989], expressed in a somewhat different form. These authors also illustrate how conclusions about $qdir2$ can be exploited by QSIM to prune admissible state transitions. Although stated in terms of M^δ , the proposition can be generalized for multivariate functions defined by M^Δ .

In addition, the qualitative second derivatives δ_{ii} have implications for the operation of consolidating adjacent M^Δ constraints. For further discussion of this point, see [Wellman, 1990a].

8 Conclusion

As others have pointed out [Dangelmaier, 1989; Kokar, 1987], qualitative modeling needs a general mechanism for multivariate functions, and partial derivatives provide the natural concepts for extending existing methods. In this article, I have developed such a mechanism within the framework of qualitative simulation, providing specific QSIM rules for reasoning about admissible states and state transitions subject to constraints on monotonic relationships and synergy properties of multivariate functions.

Incorporating multivariate monotonicity in qualitative simulation offers several important advantages. First, M^Δ is a more accurate reflection than *ADD* of the constraint enforceable on ordinal variables. Although it is weaker, and therefore extends the class of ODEs covered by a given QDE, it yields the same set of qualitative behaviors. Supplying the more general constraint obviates the need to make restrictive assumptions merely to express a multivariate relation. For example, it enables a more faithful QDE rendering of QPT.

Second, an approach based on properties of partial derivatives is extensible to higher-order interaction concepts (in fact it is prerequisite to a general qualitative treatment). For example, additivity can be expressed as the boundary case of second-order mixed partial derivatives, and can be enforced by analysis of collections of corresponding values. The stronger constraint can also be exploited when more precise information is available, such as numeric bounds [Kuipers and Berleant, 1988]. Supplying both first- and second-order constraints allows the reasoner to take advantage of whatever linearity holds, without imposing the all-or-none requirement typically present in automated dynamic analysis.

Substantive filtering based on M^Δ constraints requires that a sufficient sample of corresponding values be available. In particular, information about the impact of some set of perturbations is used to constrain other perturbations. This suggests that the qualitative synergy constraints may have particular value for analysing incremental change, perhaps more than for analysis of absolute dynamic behavior. Indeed, preliminary investigation suggests that second-order conditions justify rules for comparative dynamic analysis complementary to those assembled by Weld [1988]. Properties related to synergy have proven useful for comparative statics (equilibrium analysis) as well.

The emphasis on differential behavior also suggests the potential application of these methods to the problem of qualitative control. In particular, the relation of supermodularity to the monotone decision property provides a basis for deriving qualitative properties of optimal control strategies for systems described with QDEs. For some examples and a discussion of this approach within a decision-theoretic context, see [Wellman, 1990b].

A complete account of the potential uses of M^Δ and $M^{\Delta\Delta}$ awaits further research along these lines. The analysis presented here provides a starting point for a fuller exploration of these and other concepts of monotonicity and qualitative synergy.

A Proofs

Proposition 1 *A qualitative state is qdir-consistent with respect to $M^\Delta(x_1, \dots, x_n, y)$ iff the following conditions are satisfied:*

1. $\forall i [qdir(x_i) = std] \rightarrow qdir(y) = std$.
2. $\forall i [qdir(x_i) = std \vee (qdir(x_i) = inc \wedge \delta_i = +) \vee (qdir(x_i) = dec \wedge \delta_i = -)]$
 $\wedge \exists j [qdir(x_j) \neq std] \rightarrow qdir(y) = inc$.
3. $\forall i [qdir(x_i) = std \vee (qdir(x_i) = dec \wedge \delta_i = +) \vee (qdir(x_i) = inc \wedge \delta_i = -)]$

$$\wedge \exists j [qdir(x_j) \neq std] \rightarrow qdir(y) = dec$$

PROOF (only if): First, note that the possible qdirs of y are mutually exclusive. Next, if the antecedent of condition 1 is satisfied, then all the summed terms on the right-hand side of (1) are zero, and so must be the left-hand side, which defines $qdir(y) = std$. If the antecedent of condition 2 [3] is satisfied, then all the summed terms on the right-hand side of (1) are nonnegative [nonpositive], and at least one is strictly positive [negative]. Therefore, the sign of dy/dt , and hence $qdir(y)$, is unambiguously determined.

(if): First, note that any combination of qdirs for the x_i alone is consistent with a particular M^Δ constraint. The antecedents of the three conditions are mutually exclusive, although not exhaustive. If both the antecedent and consequent of one of the conditions are satisfied, then $qdir(y)$, is unambiguously determined, thus the state is qdir-consistent. If none of the antecedents are satisfied, then either there is a term on the right-hand side of (1) that is ambiguous ($\delta_i = ?$), or at least one of the terms is strictly positive, and one strictly negative. In either case, because the magnitudes are completely unconstrained by qdir and constraint, the sum might have any sign, and hence any $qdir(y)$ is consistent. \square

Proposition 2 *Suppose (a_1, \dots, a_n, b) is a cvt for variables (x_1, \dots, x_n, y) . An assignment to the variables is consistent with respect to the cvt and constraint $M^\Delta(x_1, \dots, x_n, y)$ iff there is some i such that $\delta_i = ?$, or the following conditions are satisfied:*

1. $\forall i [x_i = a_i] \rightarrow y = b$.
2. $\forall i [x_i \geq_{\delta_i} a_i] \wedge \exists j [x_j \neq a_j] \rightarrow y > b$.
3. $\forall i [a_i \geq_{\delta_i} x_i] \wedge \exists j [x_j \neq a_j] \rightarrow y < b$.

PROOF Note that despite the difference in form, the conditions of this proposition are analogous to the previous one, with the ordinal relation with corresponding values substituted for the qdir.

(only if): Violation of the first condition directly contradicts the premise that $y = g(x_1, \dots, x_n)$ and $b = g(a_1, \dots, a_n)$. If the antecedent of condition 2 [3] is satisfied, then the assignment differs from the corresponding values a_i along every dimension (at least one of them strictly) in the direction of increasing [decreasing] g . Hence, to be compatible with the cvt, y must also vary in that direction.

(if): See the proof of Proposition 3, which subsumes this result. \square

Proposition 3 *Let CV be a set of cvts consistent with a constraint $M^\Delta(x_1, \dots, x_n, y)$. Suppose a qualitative state satisfies the conditions of Proposition 2 for this constraint and each cvt in CV . Then for any point covered by this qualitative state, there exists a*

continuous function g that obeys the M^Δ constraint, satisfies all cvts in CV , and includes this point.

PROOF Assume for the nonce that $\delta_i \neq ?$, $1 \leq i \leq n$; the general case can be handled by a simple modification of the following argument. The set of possible values of (x_1, \dots, x_n) can be interpreted as a Euclidean vector space, with an associated partial order \gg_Δ defined as $(x_1, \dots, x_n) \gg_\Delta (x'_1, \dots, x'_n)$ iff $x_i \geq_{\delta_i} x'_i$ for $1 \leq i \leq n$ and there exists a j , $1 \leq j \leq n$ such that $x_j >_{\delta_j} x'_j$. Each cvt specifies a point in this vector space, and by the total ordering of y 's landmarks, orders these vectors in a manner compatible with \gg_Δ (by hypothesis, CV is consistent with M^Δ). A qualitative state consistent with every cvt represents another point in this space, totally ordered with respect to the others and consistent with \gg_Δ . The proposition follows from the following fundamental result.¹⁵

Lemma 7 Let F be a finite set of points in a Euclidean vector space R^n , and let $\gg_{\Delta, F}$ be a partial order that extends \gg_Δ and totally orders F . Then there exists a continuous $\gg_{\Delta, F}$ -preserving function $g : R^n \rightarrow R$.

In our case, F consists of the points in CV plus an arbitrary point in the given qualitative state. The function g has the desired properties, establishing that our constraint filter is complete for the full set of cvts. Finally, we note that this argument can be modified to cover any Δ by defining the vector space as the projection of (x_1, \dots, x_n) that excludes the dimensions i where $\delta_i = ?$. Any extension of the function g that behaves as above when these ambiguous dimensions are fixed satisfies our conditions. \square

Proposition 4 The QSIM constraint $ADD(x_1, x_2, y)$ as described by Kuipers [1986, Appendix B] admits the same qualitative state transitions as $M^{(+,+)}(x_1, x_2, y)$.

PROOF The admissible state transitions are defined by applying the constraint filter to the set of topologically possible transitions from a given state. We show that QSIM's constraint filter for ADD is equivalent to the filter defined by Propositions 1 and 2 for the special case $n = 2$ and $\delta_1 = \delta_2 = +$. For qdir-consistency, we note that the conditions of Proposition 1 for this special case are exactly equivalent to the table of Kuipers, Section B.2.1. For consistency with cvts, we observe that Proposition 2 excludes all transitions ruled out by Kuipers, Propositions B.4–B.8 and their symmetric counterparts. Because the $M^{(+,+)}$ filter is sound, it cannot prune more than ADD . \square

Proposition 5 Let B, A_1, \dots, A_n be decreasing sequences of values for variables y, x_1, \dots, x_n , as defined above, for the cvt collection $CV' = CV \cup \{cvt(s)\}$.

¹⁵The lemma is due to J. D. Mashburn, personal communication (report in preparation).

State s is inconsistent with CV and the constraint $M^\Delta(x_1, \dots, x_n, y)$ if there is no ordered assignment to $b^1 > \dots > b^{|B|}$ satisfying the inequalities formed by instantiating the schema (4) for all $cvt \in CV'$, for $1 \leq i < j \leq n$, $1 \leq k \leq |A_i| - 1$, and $1 \leq l \leq |A_j| - 1$.

PROOF Since the stated inequalities are a direct translation of the M^Δ constraint, the corresponding values, and s , their unsatisfiability entails that these elements are incompatible. If M^Δ and CV are jointly consistent, then s may be pruned. (To strengthen this to a completeness result, we would have to establish that the assignment to the b^i implies the existence of a continuous function over the whole domain that fits the assignment and preserves the supermodularity condition. In other words, we need an analog to Lemma 7 for supermodular spaces.) \square

Proposition 6 Given a bivariate monotonicity constraint $M^\delta(x, y)$ and an associated second-order constraint $M^\Delta(x, y)$, where Δ has only a single element, δ^2 ,

$$qdir2(y) = \delta^2 \oplus (\delta \otimes qdir2(x)), \quad (6)$$

where \oplus and \otimes denote sign addition and multiplication, respectively.

PROOF $qdir2(y)$ is determined by the sign of y 's second time derivative.

$$\frac{d^2y}{dt^2} = \frac{d^2y}{dx^2} \left(\frac{dx}{dt} \right)^2 + \frac{dy}{dx} \frac{d^2x}{dt^2}. \quad (7)$$

The terms on the right-hand side of (7) correspond directly to those of (6). The sign of d^2y/dx^2 is given by δ^2 , the second (squared) term is always positive, dy/dx has the sign δ , and d^2x/dt^2 the sign $qdir2(x)$. See also Kuipers et al. [1989, Proposition 5]. To generalize this result for multivariate functions (i.e., with Δ substituting for δ), replace the d/dx terms by partial derivatives. \square

Acknowledgments

I am grateful to Benjamin Kuipers and the Qualitative Reasoning Group at the University of Texas for providing QSIM for my vivisection experiments. Joe Mashburn provided valuable assistance in the proof of Proposition 3. Thanks are due also to Heidi Danielmaier, Jon Doyle, Benjamin Kuipers, Elisha Sacks, and Dan Weld for discussions on qualitative simulation and related topics addressed in this paper.

References

- [Crawford et al., 1990] James Crawford, Adam Farquhar, and Benjamin Kuipers. QPC: A compiler from physical models into qualitative differential equations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 365–372. AAAI, 1990.

- [Dangelmaier, 1989] Heidi Therese Dangelmaier. A qualitative representation for manipulator kinematics and other vector and scalar fields. Master's thesis, University of British Columbia, 1989.
- [Forbus, 1984] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.
- [Forbus, 1989] Kenneth D. Forbus. The qualitative process engine. In Weld and de Kleer [1989].
- [Kokar, 1987] Mieczyslaw M. Kokar. Critical hypersurfaces and the quantity space. In *Proceedings of the National Conference on Artificial Intelligence*, pages 616–620. AAAI, 1987.
- [Krantz *et al.*, 1971] David H. Krantz, R. Duncan Luce, Patrick Suppes, et al. *Foundations of Measurement*. Academic Press, New York, 1971.
- [Kuipers, 1984] Benjamin Kuipers. Commonsense reasoning about causality: Deriving behavior from structure. *Artificial Intelligence*, 24:169–204, 1984.
- [Kuipers, 1986] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.
- [Kuipers and Berleant, 1988] Benjamin Kuipers and Daniel Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 324–329. AAAI, 1988.
- [Kuipers *et al.*, 1989] Benjamin Kuipers, Charles Chiu, David T. Dalle Molle, et al. Higher-order derivative constraints in qualitative simulation. Technical Report AI89-117, AI Laboratory, University of Texas, Austin, TX, October 1989.
- [Michelena and Agogino, 1989] Nestor Michelena and Alice Agogino. Deterministic monotonic influence diagrams. Working Paper 89-1101-1, Berkeley Expert Systems Technology Laboratory, December 1989.
- [Papadimitriou and Steiglitz, 1982] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [Ross, 1983] Sheldon M. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, 1983.
- [Sacks, 1990] Elisha P. Sacks. A dynamic systems perspective on qualitative simulation. *Artificial Intelligence*, 42:349–362, 1990.
- [Weld, 1988] Daniel S. Weld. Comparative analysis. *Artificial Intelligence*, 36:333–373, 1988.
- [Weld and de Kleer, 1989] Daniel S. Weld and Johan de Kleer, editors. *Readings in Qualitative Reasoning About Physical Systems*. Morgan Kaufmann, 1989.
- [Wellman, 1990a] Michael P. Wellman. Exploiting functional dependencies in qualitative probabilistic reasoning. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 2–9, Cambridge, MA, 1990. To be reprinted in Piero P. Bonissone, editor, *Uncertainty in Artificial Intelligence 6*. North-Holland, 1991.
- [Wellman, 1990b] Michael P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44:257–303, 1990.
- [Wellman and Henrion, 1991] Michael P. Wellman and Max Henrion. Qualitative intercausal relations, or Explaining “explaining away”. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, 1991.

The Ontology and Representation of Situations*

Robert Wilensky
 Division of Computer Science
 University of California, Berkeley
 Berkeley, CA 94720

Abstract

The relation between sentences and what they describe is an important theoretical and practical representational issue. One issue of concern is the ontology of situations, which has generally been too limited. One way to overcome these limitations is to introduce an operator that associates a situation with any logical sentence, as advocated by Schubert and Hwang (1989). Unfortunately, such an approach appears to introduce serious difficulties. Instead, I suggest that the best solution is to simply extend the usual ontology of situations somewhat, while maintaining a separate set of entities that correspond to propositions. The components of this proposal are by and large familiar. However, considering them together gives a somewhat novel picture. For example, it seems that having objects that designate intensions becomes unnecessary. Some of the implications of an extended ontology of situations are explored, including the relation of propositions to situations and the notion of equality of situations.

1. Introduction

The relation between sentences and what they describe is an important theoretical and practical representational issue. This issue comprises two primary components: An ontological issue about what the major classes of entities to be represented are, and a formal issue of how best to represent them. Some systems of representation

make a point of providing entities corresponding to propositions, and some provide entities corresponding to situations, although the distinction between the two is sometimes confused. Both propositions and situations are useful, but for quite different purposes. Also, the representation of situations and propositions can be done in various ways, some of which have important representational advantages.

Situations on the whole have been relatively neglected. When they are provided at all, the ontology of situations has generally been too limited. One way to overcome these limitations is to introduce an operator that associates a situation with any logical sentence, as advocated by Schubert and Hwang (1989). Unfortunately, such an approach appears to introduce serious difficulties.

Instead, I suggest that the best solution is simply to extend the usual ontology of situations somewhat, while maintaining a separate set of entities that correspond to propositions. The components of this proposal are by and large familiar. However, considering them together gives a somewhat novel picture. For example, it seems that the notion of objects designating intensions becomes unnecessary. Some of the implications of an extended ontology of situations are explored, including the relation of propositions to situations and the notion of equality of situations.

2. Background

In this section I review some of the issues that have been used to motivate certain representational choices. I believe most of this material well-known. However, I review and elaborate on it here to emphasize what is at stake in these representational decisions, and what problems remain with various approaches currently in use.

Many theories of natural language presume that sentences describe events, states or processes. (Events, states and processes have been collectively called "eventualities" by Bach (1983); the term "situation" has also become popular in this context (Barwise and

*This research was sponsored by the Defense Advanced Research Projects Agency (DoD), monitored by Space and Naval Warfare Systems Command under Contract N00039-88-C-0292 and by the Office of Naval Research, under contract N00014-89-J-3205. I would like to thank Peter Norvig, Pat Hayes and Len Schubert for helpful comments that contributed to this work. Nigel Ward, Paul Jacobs and Stuart Shapiro provided many detailed comments on and corrections to a previous version of this paper.

Perry, 1983). I will use the latter term informally here as a superordinate category encompassing actions, events, states, processes, and whatever else sentences may refer to, but without importing any particular theoretical baggage.) For both theoretical and practical reasons, it is useful to conceive of situations as entities to which one may refer or about which one may say things. On the other hand, sentences are purported to have a logical form of a predicate-argument nature, with the matrix verb serving as the predicate and its complements specifying the arguments. However, this logical form is propositional, and does not readily provide an entity to designate the underlying situation.

For example, consider a simple sentence like the following:

(1) Jan gave Lynn a cigar.

The logical form commonly attributed to (1) is some variant of the following:

(2) Give(jan, lynn, cigar1)

(where temporal information and the facts that "jan" and "lynn" are people and "cigar1" a cigar are ignored for the time being). Now suppose (1) were followed by

(3) This made Terry furious.

We would like some way to say that the pronoun "this" refers to the event described in (1). However, there is no readily apparent way of expressing this fact given the representation of (1) as (2). The problem is that while (2) expresses the logical structure of the sentence, it does not provide any object designating the reported event. We might be tempted to have the predicate corresponding to the verb "make" take a proposition as an argument, and then represent (3) as the following:

(4) Make(Give(jan, lynn, cigar1), furious)

But this approach doesn't help, because we have no way of asserting that this was the same event as the one described in (1), and not some other, albeit similar, occurrence.

A second motivation for having entities corresponding to situations is to capture important generalizations across varying linguistic forms. For example, consider the following utterance:

(5) Jan giving Lynn a cigar made Terry furious.

Here the phrase "Jan giving Lynn a cigar" has a quite

similar content to that of (1). However, the stance of a speaker using this sentence to its content is different: Both sentences describe an event of Jan giving Lynn a cigar, while the speaker of (1) is asserting the occurrence of the event, and that of (5) presupposing it (or, as Schubert et al. (1979) would say, the *propositional content* is the same, while the *pragmatic aspects* differ). Whatever representation we choose should make this similarity of content apparent, while allowing us to express the difference in attitude taken toward this event.

This need becomes even more apparent when we consider lexical nouns that encode events (Parsons, 1985). Some nouns, like "punch" and "destruction", seem to describe the same events as certain verbs, e.g., "punch" "destroy". This claim is supported by the observation that sentences like the following appear to have the same truth conditions, if we adjust for presupposition:

- (6) The Romans destroyed the city, resulting in much grief.
- (7) The destruction of the city by the Romans resulted in much grief.

Whatever meaning difference one can discern between the noun and the verb does not appear to be attributable to a difference in the events they encode. Hence the representational reflex of attributing a predicate to a verb and an argument position to a complement would pave over a crucial and obvious generalization.

A third argument for representing the designatum of a sentence as an entity is a corollary of the first two. Namely, there are often a number of different things to say about a situation, and we need some way of saying only some of them, and, of saying additional things about that situation later on. In particular, elements of a sentence frequently specify the time, place and manner of a situation, as well as its modality and the speaker's attitude toward the situation. Some of these may be thought of as higher-order predicates. However, such an approach is problematic in some cases.

In particular, consider the problems that arise from distinguishing adjuncts from complements. Many proposals attribute argument positions to a predicate in accordance with the complement structure of the verb associated with that predicate; adjuncts are dealt with in some other manner. Thus, in (1) above, we presumed that the predicate corresponding to the verb "give" is a three-place predicate because this use of the verb takes three complements. However, suppose we had continued the sentence with "in New York". Since this phrase is an adjunct, we would not have provided an argument posi-

tion for it. Indeed, such a solution would not easily be possible, since an arbitrary number of such adjuncts can be present. Moreover, the adjunct seems to predicate something about the whole event, rather than constitute a component of it, as was the case for the complements.

We can accommodate such adjuncts into the representation by introducing a higher-order "in" predicate, one of whose arguments would be the predication describing the entire situation (or, alternatively, the application of some functional operator to this formula, denoting the situation) and the other, the location. However, this proposal is awkward for many reasons, not the least of which is that we would then have at least two semantically unmotivated but formally distinct "in" predicates, one which applies to formulas, and the other to individuals (cf. Shapiro 1971b).

The situation is compounded by cases in which the adjunct does not appear to apply to the entire situation. For example, consider the following sentences:

- (8a) Jan stumbled into the room.
- (8b) Jan threw up in the sink.
- (8c) Jan bled on a shirt.
- (8d) Jan sliced the salami onto the bread.

Note first that treating the prepositional phrases as complements appears untenable. If we did so, then we must make one of two assumptions: We could assume that each verb corresponds to a predicate with one more argument than had been obvious, and that this argument is omitted most of the time. However, this proposal would lead to the wrong analysis of at least some cases; for example, Jan stumbling doesn't require that Jan stumbled directionally. Alternatively, we can assume that each verb corresponds to several different predicates (e.g., "bleed" and "bleed-directionally", and "throw-up" and "throw-up-directionally"). But this proposal requires a gratuitous meaning postulate to sanction the inference from the n+1-place predicate to the n-place predicate (e.g., to sanction "Jan bled" from "Jan bled on a shirt"). Both alternatives seem unmotivated, at best.

Thus, we must consider the prepositional phrases in (8) to be adjuncts. However, if adjuncts are not to be incorporated in the associated predicate, we are left in the following situation. We can once again attempt to treat the adjunct as a higher-order predicate. For example, we might represent (8c) as the following:

- (9) Onto(Bleed(jan),shirt1)

But then prepositions like "into" and "onto" get a radically different treatment when they are complements

from when they are adjuncts, even when there is no semantic justification for doing so. This situation is worse than the case above in which the prepositional phrase modified the whole sentence, because here, there is presumably no difference at all in meaning between the use of the prepositional phrase in sentences like (8d) and those in which it is a complement, such as the following:

- (10) Jan threw the ball onto the roof.

In sum, when the distinction between being a complement and an adjunct does not correspond to a distinction in meaning, but is merely a fact about whether or not a verb happens to subcategorize for a particular type of constituent, then the difference should not be reflected in the underlying logical form. But doing so appears to be problematic if the designatum of the matrix verb and complements are thought to correspond to a predication.

Note that a version of this problem applies whenever we have the quite common case of a verb that can describe the same situation with several different complement structures. For example, consider the following sentences:

- (11a) Jan opened the jar.
- (11b) Jan opened the lid (to the jar).
- (12a) Jan tied the string.
- (12b) Jan tied a knot (in the string).

If we proposed having a predicate per valence description, we would once again require a meaning postulate to recognize that the (a) sentences follow from the (b) sentences. Thus, one goal of whatever representation we adopt should be to do without an additional postulate when the inference is analytic.

Before going on, note that the various problems just presented are essentially invariant with the other choices one can make about representation. For example, if one believes that the meaning of "give" should be decomposed into primitives, and that only the latter can appear in the description of events, then all the same problems persist.

3. Proposed Solutions

To summarize, the problem is to determine what representations we can use for the situations described by utterances so that proper generalizations can be captured. In particular, the grammatical form of lexical items should not have an undue influence, lest we should be incapable of capturing generalizations that hold between various parts of speech or about terms that

admit various complements and adjuncts.

3.1. Situation-based Approaches

A well-known solution to some of the problems mentioned above was proposed by Reichenbach (1947) and elaborated by Davidson (1969,1980). Basically, this amounts to postulating another argument for every event predicate. According to this account, the logical form of (1) includes not (2) but rather

$$(13) \text{Exists } x (\text{Give(jan,lynn,cigar1,}x\text{)})$$

where x is construed as an event. We will discuss extending this sort of analysis to states below, but for the time being we will assume that such an extension is straightforward.

This particular formalism can be characterized as "situation-based" and "predicate-based". By situation-based, I mean that the system provided entities corresponding to individual situations (e.g., the variable in (13)). By "predicate-based", I mean that the situations are specified by altering each domain predicate in some fashion, in this case, by the inclusion of an additional argument. An example of a use of such a notation is Hobbs (1986).

It is the situation-based character of the formalism that provides a solution to some of the concerns voiced above. If we skolemize (13) to produce a constant x_1 , say, then we have our unique referent. We could postulate that the logical form corresponding to (3), with the referent of the pronoun established, was not (4) but rather the following:

$$(14) \text{Make}(x_1,\text{furious})$$

Similarly, "Jan giving Lynn a cigar" might be represented as x_1 with the same formula predicated about it as in (13). In other words, the "giving" event described in (1) and (5) has the same logical form, although (1) asserts that this event transpired while (5) presumes it.

Also, it is trivial to add in first-order format information about the event. For example, specifying the proper time might be done by something like the following:

$$(15) \text{Before}(x_1,\text{NOW})$$

Similarly, the logical form of verbal adjuncts is easy to accommodate in such a formalism. For example, the sentence

$$(16) \text{Jan gave Lynn a cigar in New York.}$$

might be construed as follows:

$$(17) \text{Exists } x (\text{Give(jan,lynn,cigar1,}x\text{)} \\ & \& \text{In(x,newyork)})$$

3.2. Aspectualized Representations

These last two observations point to some degree of arbitrariness in deciding on the argument structure of various predicates, to which the situation-based notation offers an attractive solution. Instead of having to predetermine a fixed number of arguments for a given predicate, we assume that each predicate has only one argument, namely, the situation. Other aspects of the situation (for example, if it were an action, who its actor is) are added by additional predication completely analogously to the way time and location were added above. In this formulation, (1) is given the logical structure

$$(18) \text{Exists } x (\text{Give}(x) \& \text{Agent(jan,}x\text{)} \\ & \& \text{Patient(cigar1,}x\text{)} \\ & \& \text{Recipient(lynn,}x\text{)})$$

In (18), the argument positions of the main predicate are exchanged for a set of two-place predicates. This style of representation is called "slot-assertion notation" by Charniak and McDermott (1985). Indeed, one may view the use of these predicates as a notational variant that allows us to make a predication with a variable number of arguments (with more than one possibly being in the same argument "position"). I will use the informal term *aspectual* (Wilensky 1986) for any two-place predicate introduced into a representation for the purpose of separating the assertion of an aspect of a situation from the assertion that something is a particular kind of situation. Aspectuals are meant to include the arcs of some semantic network systems (in particular, those arcs emanating from nodes denoting situations) and the slots of most frame-based systems.

The particular aspectuals used in the example above are the familiar relations imported from case theory. But nothing in our argument depends on the choice of these predicates, although one must eventually specify a set that is appropriate for the particular situation involved. Some theories assume only a small set of quite general aspectuals; others prefer a much larger set that depends on the situation, for example, "Giver", "Givee", "Given" for "Give" situations. We make no commitment here, and use whatever aspectuals are useful for expository purposes.

With this representation, verbal complements can now be given the same logical treatment as verbal adjuncts: Both are represented by predication using aspectuals. E.g., extending (18) to incorporate "in New York" involves adding a predication that looks formally identical to those corresponding to the complements.

Representations for non-sentential verbal adjuncts can be given analogously. For example, (8c) above could be represented as the following:

- (19) Exists x,y,z (Bleed(x)
 & Experiencer(jan, x)
 & Patient(y,x) & Blood(y)
 & Onto(y,z) & Shirt(z))

Representing the adjunct requires only the addition of the last two predication, and the representation of the sentence without the adjunct is identical to (19) without these predication*.

Thus, such a notation allows us to represent adjuncts and complements in a logical structure that captures what is being expressed, as opposed to how these elements are packaged into the language. The latter difference is maintained in the grammar, where it belongs, but not in the logical form.

Note also that the resulting notation treats situations analogously to physical objects, persons, etc. For example, had our representation of (1) been more elaborate, and included information about the nature of the entities corresponding to the noun phrases in the sentence, it might have been more like the following:

- (20) Exists $x,jan,lynn,cigar1$
 (Give(x) & Agent(jan, x) & Person(jan)
 & Patient(cigar1, x) & Cigar(cigar1)
 & Recipient(lynn, x) & Person(lynn))

Here "Person(jan)" denotes the fact that Jan is a person analogously to "Give(x)" denoting that x is a giving event. In both cases we would have to make further predication to express additional facts about the nature of the entities. For example, we would need an additional predication to express that the first name of the person Jan is "Jan", just as we needed an additional predication to express that the patient of the giving

*One might argue for a more elaborate analysis of "onto" both for (8c) and (10). For example, another plausible analysis is that there are events described by the sentences (bleeding and throwing, respectively) and that these caused additional events (blood moving onto a shirt and a ball moving onto a roof). The important point is that parallel analyses are available for the two sentences using either analysis.

event was a cigar.

This uniformity is particularly important in capturing the unity of a preposition like "in", when we have the intuition that the same spatial meaning is being used regardless of the arguments. Thus, the representation of the phrase "the pen in the box" can make use of exactly the same "in" as that of "The pen broke in the box". Both have the following logical form:

- (21) In($x,box1$)

where " x " is alternatively a pen, or the event of a pen breaking.

Finally, the analytic inferences involving verbs with different valence structures can be handled appropriately. We specify the structure of the underlying event by specifying the set of aspectuals it accommodates. Verbs with multiple valences hence may specify the same event, but supply the arguments to different aspectuals. An event-class corresponding to the verb "open", e.g., might have aspectuals for the object constituting a barrier or container, the object moved to create the opening, etc. Then the representations for "open the jar" and "open the lid" could be identical except for the aspectuals that the uses of the verb specify; i.e., the first would relate an opening event to a jar via a "containing-object" aspectual, and the second would relate an opening event to a lid via a "object-moved" aspectual. Similarly, in the case of optional complements, when a complement is omitted, the representation is identical except for the omission of a predication using the corresponding aspectual.

Note that the key property of the representation that solves many problems for us is that it is amenable to "aspectualization", i.e., we can predicate different aspects of a situation separately. A situation-based, predicate-based system does not necessarily have this property, as we noticed when a predicate provides argument positions corresponding to complements. However, a situation-based, predicate-based system in which domain predicates take only situations as arguments is aspectualized. Henceforth, when I refer to predicate-based representations, I will assume they are aspectualized, unless otherwise noted.

3.3. Category-based Approaches

Another way to have representations that are amenable to aspectualization is to have within one's system entities corresponding to categories, including categories for situation types, e.g., "Giving" and "Event", as well as "Person" and "Book". We would also intro-

duce a predicate, say AIO (An Individual Of) relating an individual to a category it belongs to. Then, instead of (20) we would have the following:

- (22) Exists x , jan, lynn, cigar1
 (AIO(x , Giving)
 & Agent(jan, x) & AIO(jan, Person)
 & Patient(cigar1, x) & AIO(cigar1, Cigar)
 & Recipient(lynn, x) & AIO(lynn, Person))

Such formulas, or perhaps, the skolemized versions of them, can be viewed as the basis for many semantic network and "frame-based" systems, in particular, of all those formalisms that have some explicit relation meaning "an individual of". I will call a proposal in which there are individuals corresponding to categories of situations a "category-based" representation*. I would construe Simmons (1973), Rumelhart, Lindsay and Norman (1972), Hendrix (1975), Schank (1975), Bobrow and Winograd (1977), Roberts and Goldstein (1977) and Brachman (1979) as all adopting category-based schemes of some sort.

This classification is akin to what Israel (1983) calls "inheritance-based networks". However, the above systems vary in the degree to which inheritance per se is prominent in them; for example, inheritance is rather conscientiously avoided in Conceptual Dependency (Schank, 1975), while it is quite prominent in KRL (Bobrow and Winograd, 1977) and KL-ONE (Brachman, 1979). What all these systems do have in common is the idea of an individual of a category corresponding to a situation type.

One advantage of category-based representations is the availability of categories for predication. For example, the representation for a classic sentence like "The Saber-toothed tiger is extinct" could involve a reference to the concept "Saber-toothed-tiger". However, I will not make a substantial case here for the advantages of category-based over predicate-based representation,

*Allen (1987) interprets the categories of semantic networks as corresponding to logical predicates rather than individuals. Perhaps this is because the semantics of the terms corresponding to categories might be problematic for some, who like to interpret formulas over models that somehow correspond to the world (what I have called a "direct correspondence" theory (Wilensky, 1986)). In this case, "Giving", etc., probably have to be construed as sets. However, semantic network theorists (e.g., Maida and Shapiro (1985)) often prefer to interpret formulas with respect to some normative cognitive agent (a "cognitive correspondence theory"). In this case, "Giving" is supposed to be the concept of giving; each individual of such a category denotes a conception of an individual event; the actual events, etc., are generally not designated at all.

except to note that an aspectualized representation seems inherent in the category-based representation, whereas a predicate-based approach must be tailored to have this property.

Some semantic networks make a habit of providing categories, but no individuals corresponding to situations. Rather, in these formalisms, the emphasis is on nodes representing propositions. For example, Cercone (1975), Schubert et al. (1979) and especially Shapiro (1979) fall into this category. Such systems are sometimes called "propositional semantic networks", although this term seems to be used to emphasize the point that, in such a system, every accessible assertion is a node (Maida and Shapiro, 1985), rather than as a contrast between systems that have nodes for propositions versus those that have nodes for situations. To be sure, such systems are not necessarily hostile to the inclusion of individuals corresponding to situations; they just do not make a commitment to provide such individuals. While having individuals corresponding to propositions is a useful idea we will return to later, it does not address the problem of providing referents for situational anaphora.

4. A Problem

Situation-based representations provide objects corresponding to situations; these are convenient when subsequent utterances refer to previously mentioned events. However, an important characteristic of situation-based proposals is that their vocabulary is generally *lexically determined*. That is, the vocabulary of predicates or categories is approximately that of the word senses of the various verbs, etc., recognized as designating situations. However, as Schubert and Hwang (1988) point out, such representations do not yield an object for sentences involving logical operators or quantifiers, but only for atomic formulas. They give as an example the following sentence:

- (23) Everyone looked at Mary.

They note that standard situation-based representations would provide no object for the event described by this sentence, even though such an event may be referred to subsequently, if followed, for example, by a sentence like (3) above:

- (3) This made Terry furious.

Schubert and Hwang propose a solution to this problem in the form of *episodic logic*, in which symbols designating situations are related to propositions capturing the logical structure of the sentence via connectives

introduced just for this purpose. In particular, the connective ‘**’ relates a proposition to a situation it “characterizes”. For example, (23) would be represented* as

- (24) Exists e1 (**(All x (Person(x)
-> Look-at(x,mary)),e1))

Here “e1” is the situation with characterization “Everyone looked at Mary”. This gives us a handle on the entire situation, just as the word-style representations did on events requiring only atomic formulas. For example, we can represent the following sentence

- (25) This made her blush.

as describing the situation

- (26) Exists e2 (Cause(e1,e2) & **(Blush(mary),e2))

allowing the scope of the existential in (24) to extend a bit further than the parenthesis would ordinarily allow.

Schubert and Hwang also propose a more fundamental operator, “*”, that is used to relate a situation to a partial description. To illustrate the significance of characterizing versus partial descriptions, they note that, if it were everyone looking at Mary *desirively* that made her blush, then (24) with “**” replaced with ‘*’ would still be true, even though (23) together with (25) would not be.

I call Schubert and Hwang’s proposal a “proposition-based-situation” proposal. That is, they define situations as objects that correspond to logical propositions. Note that what I am calling a proposition-based-situation solution has little to do with what has sometimes been called a “propositional semantic network”, which Maida and Shapiro (1985) describe as a network “in which *every* assertion that can be stored or accessed ... is ... represented by a node”. Rather, I use the term to refer to proposals in which situations are defined by an operator that relates them to a logical formula.

Schubert and Hwang’s proposal is very attractive because it allows one to have situational objects corresponding to arbitrary propositions. However, it is not unproblematic. Significantly, we are back to a system in which complements and adjuncts are treated quite differently. Schubert and Hwang represent verbal

*These authors use the terms “event”, “situation” and “episode” interchangeably. Also, I have omitted temporal information here which the authors include in their analyses. Finally, Schubert and Hwang use an infix notation I have eschewed for uniformity’s sake.

complements as arguments to the corresponding predicate, but advocate having temporal modifiers that take episodes as arguments. Thus, they use

- (27) Before(e1,NOW)

to express the fact that episode e1 occurred in the past. And, as in the example above, causal relations hold between episodes. But then the representational implications of being a complement versus an adjunct, and all the associated problems discussed above and conveniently eliminated by either predicate- or category-based representations are reintroduced. That is, the representation is not aspectualized.

Even worse, it is not clear that the semantics of “**” and “*” can be made coherent. In particular, what it means to “characterize” a situation seems problematic. Schubert suggests* that the situation described by

- (28) Looking down the barrel of the gun, John fired the gun.

is characterized by the representation corresponding to the main clause, but not by that corresponding to the dependent clause. However, the same would then presumably apply to a sentence like the following:

- (29) John fired his gun looking down the barrel.

But this sentence could be followed with

- (30) This enabled him to focus clearly on the target.

This sentence is problematic since we have no event characterized by “looking down the barrel”, and only such events are supposed to be the bearers of useful causal information.

Indeed, the whole purpose of introducing the notion of characterizing an event seems to be to avoid the annoying problem of figuring out, when one event is predicated to cause another, what is really being asserted to have caused what, since an indefinite number of predictions might apply to the two events, but most of these

*Personal communication. Schubert and Hwang have revised their ideas considerably since the initial publication of their 1989 paper. In particular, “**” now means “completely describes”, i.e., that everything else one can say about an event completely described by a proposition is entailed by that proposition. This proposal is somewhat similar to one I propose below re equality of situations. However, I believe that the other difficulties of proposition-based-situation representations, except perhaps the difficulty involving causation, are still problematic for this new proposal.

are not presumed to describe factors that have any causal role. However, there appears to be no in-principle basis for deciding whether a predication is characterizing or not. In any case, events or situations are entities in the world, so their characterization couldn't matter with respect to causality.

Part of the difficulty here may be that Schubert and Hwang's notation is intended to be situational, but the objects associated with formulas are better interpreted as propositions rather than as events. That is, what the notation really allows us to do is to tag propositions for subsequent reference. But propositions and events are quite different animals. Hence a number of problems arise, for example, the need to identify which of the multiple propositions that can be given the same identifier are implicated in causal relations and the like. I will elaborate on this point below.

Some of these difficulties appear to be intrinsic. In particular, it does not appear to be possible to re-aspectualize the representation. For example, the episodic logic representation of

(31) Jan looked at Lynn.

would be

(32) Exists e (**(look-at(jan,lynn),e))

Now, suppose we tried to regain an aspectualized form by removing arguments from "look-at" and moving them to aspectual predications of "e". Doing so would yield the following form:

(33) Exists e (**(look-at(),e)
 & Experiencer(jan,e)
 & Patient(lynn,e))

But this is essentially a category-based notation, with predicates of no arguments substituting for categories and the operator "##" having the corresponding semantics to the predicate "AIO". (I explore the implications of this observation below.) Moreover, ## has lost its ability to relate an arbitrary proposition to an event; should we try to quantify over experiencers, as was the case in the original sentence (23), we would no longer have an object that corresponds to the whole event.

5. A Solution

I propose one solution to the problem of how best to represent situations, which I term an ontological solution, because it gets around the problem pointed out by

Schubert and Hwang by hypothesizing more kinds of entities. In particular, we posit entities for the situations corresponding to sentences involving logical connectives or quantifiers. For example, recall sentence (23) above:

(23) Everyone looked at Mary.

Let us posit that this sentence corresponds to a situation class called "Complex-event". "Complex-event" has exactly the same status as a category like "Giving". However, it is amenable to any number of "sub-event" predications. Then (23) could have the following characterization:

(34) Exist c1 (AIO(c1,Complex-event) &
 All x (AIO(x,Person)
 -> Exists l (AIO(l,Looking-at)
 & Actor(x,l))
 & Patient(mary,l))
 & Sub-event(l,c1)))

That is, there is a complex event that has a subevent corresponding to each individual's looking action. Each of these actions has the standard case structure. We can refer to the entire complex event by referring to c1, while preserving the desirable structure of the individual component actions. Note that it is relatively easy now to say that one of these looking actions is derisive, say, simply by creating a description of an individual event having this property, and assertion that it is a subevent of c1.

One appealing aspect of this proposal is that it captures some of aspects of the underlying events that one would want independently. That is, that events like c1 are composed of individual subevents is a fact that is useful for any number of purposes. Therefore, relying on such an ontology for a solution to the situation-as-entity problem is not simply gratuitous. Also, the same particular category, "Complex-event", would be used to represent sentences involving many other quantifiers, such as "many", "most" or "almost all", should a suitable analysis of these quantifiers be given.

One perhaps less appealing aspect of the proposal is that it at least opens the door to having ontological categories duplicating various logical connectives. For example, consider the following sentence:

(35) John didn't go to New York.

The representation for (35) might be the following:

(36) Exists n,e (AIO(n,Non-event) &
 negated-event(e,n) &

$\text{AIO}(e,\text{Going}) \&$
 $\text{Actor}(\text{john},e) \&$
 $\text{Destination}(\text{newyork},e))$

Here n denotes the non-event of John going to New York.

Having situations that are “non-events” may seem ontologically suspect (cf. Hobbs (1985) and Hirst (1989)). However, that such situations are plausible is suggested by the fact that (35) could be followed by

- (37) This caused the firm to lose an important customer.

That is, non-events seem to be as referable and as causally culpable as “real” events.

In contrast, note that there does not appear to be a need to have disjunctive situations. For example, consider the following sentence:

- (38) Either John went to New York or Bill went to Pittsburgh.

Presumably, if (38) is followed with (37), it is not some disjunctive situation *per se* that caused the loss, but one of the two disjuncts. That is, we would have to interpret these events as either John’s going to New York or Bill’s going to Pittsburgh having caused the firm to lose an important customer. Therefore, we are not compelled to enter such a category of situations into our ontology, at least by our current arguments.

Note that a somewhat different reading is available if (38) is followed by

- (39) This infuriated Mary.

There is an interpretation of (40) in which Mary’s infuriation depends only on her having learned the disjunction, and not on her learning of one or the other event. I take this as evidence of the need for propositional rather than situational objects, which are discussed further below.

Such extended ontologies are not new, but there appears to have been confusion over what they are ontologies of. For example, Hendrix (1975) has categories like “Negations”, “Implications”, and “Disjunctions” and has nodes (actually, “supernodes” denoting spaces in his partitioned semantic network notation) that are predicated to be individuals of this category. However, these categories, unlike those associated with more basic sentences, are explicitly categories of propositions. Thus, two very different notions seem to be

confiated here. “John owns a car” gives rise to a node that is an individual “Owning” situation, but “John doesn’t own a car” and “Everyone owns a car” yield nodes that are individual propositions; they are not situations at all. In effect, Hendrix has changed from a situation-based representation to a propositional one in midstream. Moreover, the ontology of situations is not extended to cover those situations corresponding to the more complex logical forms. While Hendrix is clear that his categories are categories of propositions, his lack of comment on the transition may indicate that the difference is not much appreciated. In this instance, Hendrix appears to build upon prior confusion, such as that in Woods (1975, p. 57), which construes individual nodes in case-based representations as standing for both assertions of facts and instances of events.

5.1. Propositions and Situations

Adding a full ontology of situations does not seem to reduce the need for propositional objects. There are at least two arguments for their existence. One is the need for propositional anaphora; since these arguments are similar to the ones used to motivate situational objects, we will not repeat them here, but mere offer the following notation: We introduce the operator “Prop” so that a sentence like

- (40) Lynn knows that Jan went to New York.

is given the following notation:

(41) $\text{AIO}(k1,\text{Knowing})$
 $\quad \& \text{Knower}(\text{lynn},k1)$
 $\quad \& \text{Known}(p1)$
 $\quad \& \text{Prop}(p1,\text{AIO}(g1,\text{Going}))$
 $\quad \quad \& \text{Goer}(\text{jan},g1)$
 $\quad \quad \& \text{Destination}(\text{newyork},g1))$

Here $p1$ designates the proposition that Jan went to New York.

Considering both situations and propositions together eases the representational burden that is sometimes overextended to each one. Note that as a reification of propositions, rather than situations, proposition-based-situation proposals are no longer problematic. Unlike situations, propositions are completely bounded in extent, and are thus identical given the same content. That is, there is only one proposition that Jan went to New York, and that Jan went to New York is its entire contents; however, the event of Jan’s going to New York might have any number of other components.

5.2. Intensional Representations and the Equivalence of Situations

In intensional semantic networks, e.g., Maida and Shapiro (1985), much is made of the view that no two distinct nodes are equal, and that referential opacity is the norm. However, with propositional objects, it seems that all needs for objects in the representation designating intentions can be eliminated. That is, it is perfectly acceptable to assume that situational and other individuals refer to objects in the world, and that nothing in our system corresponds to an intension.

One motivation for intensional representations (e.g., Woods (1975) and Maida and Shapiro (1985)) is that without them, it is not possible to represent sentences like "Jan knows that the Morning Star is the Evening Star". This is because such a representation would only have one node representing the extension, about which it is predicated that the object was both the Morning Star and the Evening Star. But in fact, with propositions in our notation, this is not the case. Here is one possible way to represent exactly this sentence without intentions (i.e., without representational objects designating intentions):

- (42) AIO(k1,Knowing)
 - & Knower(jan,k1) & Known(p1)
 - & Prop(p1, AIO(m1,Star)
 - & Named(m1,"Morningstar")
 - & AIO(e1,Star) & Named(e1,"Eveningstar")
 - & =(e1,m1))

That is, Jan knows that something called the Morning Star and something called the Evening Star are the same thing.

Propositional contexts are considered referentially opaque, but elsewhere we can freely substitute equals for equals and arrive at truth-conditionally equivalent formulas. While there are many variations to the analysis offered in (78), we are able to do without intentions here because we are willing to provide "redundant" information in the proposition. Thus, rather than have an intension for the Evening Star, we simply specify the propositional content of "a star named 'The Evening Star'", and associate this content with a term. We may already have a similar term that we know is called "The Evening Star", so a very similar fact is already in the knowledge base. However, this redundancy seems to correspond to a rather intuitive interpretation of the information the sentence is actually conveying.

Similar reasoning applies directly to situations. In the case in which one event logically entails the other, as

for example an event of walking entailing an event of moving, no obvious problems arise. However, consider the case in which one of the two events does not necessarily entail that other. For example, suppose that I have replaced a pipe under my sink, and that this replacement fixes a problem with my sink. In this case, we could say that one event, described as replacing a pipe, is coreferential with another event, that of fixing my sink. Suppose we attempt to represent this situation with a single entity, e1. Then we would say something like the following:

- (43) AIO(e1,Replacing)
 - & Replacer(rw,e1)
 - & Replaced(oldpipe,e1)
 - & Replacement(newpipe,e1)
 - & AIO(e1,Fixing)
 - & Fixer(rw,e1)
 - & Fixed(sink,e1)

Let us suppose further that both the replacer and the fixer are agents, and that both the thing replaced and the thing fixed are objects. That is, we can infer (44) from the first set of conjuncts of (43) and (45) from the second:

- (44) AIO(e1,Action)
 - & Agent(rw,e1) & Object(oldpipe,e1)
- (45) AIO(e1,Action)
 - & Agent(rw,e1) & Object(sink,e1)

But, if (44) and (45) are both true, we have lost some crucial information; we no longer know which aspectuals go with which event category. In particular, if we make the usual assumption that actions have unique objects, one can infer that I replaced my sink with a new pipe, which is simply false. (A version of this problem appears in Parsons (1985)).

It is probably undesirable to abandon the assumption of actions having unique objects, since then actions may have any number of objects, only one of which is qualified to be grammaticalized as a syntactic object. The better solution, it would seem, is to deny the identity of the two events. We would say instead that there was a fixing event, f1, say, and a replacing event, r1, and that the latter "comprised" the former, or that some other such intimate but non-identifying relation holds between them.

Determining the correct vocabulary of such relations is an important open problem. However, it does not appear to be unique to situations. There are at least two other related cases. One is the relation between objects and the substances that comprise them. Some approaches equate the two. For example, Lenat and

Guha (1988) define a lake as a quantity of water. However, this is certainly incorrect, as the particular water comprising the lake changes while the lake object remains constant. Another case is the relation between roles and their fillers. If we allow categories like "Passenger" along with individuals of such category, it is tempting to equate the role object with its filler. But again, problems will arise. For example, the yearly number of passengers is greater than the number of individual people who fly, since some individual may fly multiple times, and hence, count as several passengers. In both of these cases, as in the case of co-extensive but non-identical situations, the solution seems to require having a relation between objects that does not allow for substitutability, but nevertheless declares them to be co-extensive.

6. Summary

I began by acknowledging the virtues of predicate- and category-based representations, in particular, their amenability to aspectualization, but accepting Schubert and Hwang's point about such representations being limited to reifying situations described only by atomic sentences in the logic. However, we found that their proposition-based-situation proposal, while overcoming this difficulty, results in an apparently unaspectualizable representation, and hence reintroduces a number of problems, some of which predicate- and category-based representations address, plus some additional problems of its own. I abandoned hope of resolving these difficulties, and instead, sought to extend the category-based solution.

The extension is first an ontological one, in that it introduces situations corresponding to the logical forms for which category-based solutions would not otherwise apply. The problems Schubert and Hwang illuminate can thereby be resolved while maintaining category-based advantages.

The second aspect of the extension is to allow propositional objects. These are needed primarily to deal with problems of referential opacity and for propositional anaphora. Having propositions as objects in the system eliminates the need for any kind of intension altogether.

While the proposal is primarily a way of maintaining aspectualization, this necessary virtue yields a rather cumbersome representation. In particular, what used to be a simple predication now requires a variable or a constant and a number of predications; also, every utterance has both a propositional and situational object associated with it. However, this inelegance appears to be the price one has to pay for the features crucial for an

adequate representation.

7. References

- (1) Allen, James. *Natural Language Understanding*. Benjamin Cummins, Menlo Park, California, 1987.
- (2) Bach, Emmon. On Time, Tense and Aspect: An Essay in English Metaphysics. In R. Bauerle, C. Schwarze & A. von Stechow (eds.) *Meaning Use and Interpretation*. de Gruyter, New York, 1983.
- (3) Barwise, J. and Perry, J. *Situations and Attitudes*. Bradford Books, MIT Press, Cambridge, Mass., 1983.
- (4) Bobrow, D. G. and Winograd, T. An overview of KRL, a Knowledge representation language. *Cognitive Science*, Vol. 1, no. 1, 1977.
- (5) Brachman, R. J. Structure Inheritance Networks. In *Research in Natural Language Understanding*, W. A. Woods and R. J. Brachman, Quarterly Progress Report No. 1, BBN Report No. 3742. Bolt, Beranek & Newman, Cambridge, Massachusetts, 1978.
- (6) Brachman, R. J. On the Epistemological Status of Semantic Networks. In N. V. Findler (ed.), *Associative Networks: Representation and Use of Knowledge by Computers*. New York: Academic Press, 1979.
- (7) Cercone, N. Representing Natural Language in Extended Semantic Networks. Technical Report TR75-11, Department of Computer Science, University of Alberta, Edmonton, Alberta, Canada, 1975.
- (8) Charniak, E. and McDermott, D. *Introduction to Artificial Intelligence*. Addison-Wesley Publishing Company, 1985.
- (9) Davidson, Donald. The individuation of events. In N. Rescher et al. (ed.) *Essays in Honor of Carl G. Hempel*, pp. 216-234. Reidel, Dordrecht, Holland, 1969.
- (10) Davidson, Donald. *Essays on Actions and Events*. Clarendon Press, Oxford, 1980.
- (11) Hendrix, G. G. Expanding the utility of semantic networks through partitioning. In the *Proceed-*

- ings of the 4th International Conference on Artificial Intelligence*, 1975.
- (12) Hirst, Graeme. Ontological assumptions in knowledge representation. In the *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Canada, May, 1989.
 - (13) Hobbs, Jerry R. Ontological Promiscuity. In the *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago, June, 1985.
 - (14) Hobbs, Jerry R. Overview of the Tacitus Project. *Computational Linguistics*, vol. 12, no. 3, 1986.
 - (15) Israel, David. Interpreting network formalisms. In N. Cercone (ed.), *Computational Linguistics*. Pergamon Press, Oxford, 1983.
 - (16) Maida, A. S. and Shapiro, S. C. Intensional Concepts In Propositional Semantic Networks. In R. J. Brachman & H. J. Levesque (eds.), *Readings in Knowledge Representation*. Morgan Kaufmann Publishers, Los Altos, California, 1985.
 - (17) Parsons, Terence. Underlying Events in the Logical Analysis of English. In E. LePore & B. P. McLaughlin (eds.) *Actions and Events: Perspectives on the Philosophy of Donald Davidson*. Basil Blackwell, 1985.
 - (18) Reichenbach, Hans. *Elements of Symbolic Logic*. The Free Press, New York, 1947.
 - (19) Roberts, R. B. and Goldstein, I. P. The FRL Manual. Technical Report AIM-408, MIT Artificial Intelligence Laboratory. 1977.
 - (20) Rumelhart, D. E., Lindsay, P. H., and Norman, D. A. A process model of long-term memory. In E. Tulving and W. Donaldson (eds.), *Organization of Memory*. Academic Press: New York, 1972.
 - (21) Schank, R. C. Identification of Conceptualizations Underlying Natural Language. In Schank, R. C. & Colby, K. M. (eds.), *Computer Models of Thought and Language*, W. H. Freeman and Company, San Francisco, 1973.
 - (22) Schubert, L. K. and Hwang, C. H. An Episodic Knowledge Representation for Narrative Texts. In R. J. Brachman, H. J. Levesque & R. Reiter (eds.), *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pp. 444-458. Morgan Kaufmann Publishers, Los Altos, California, 1989.
 - (23) Schubert, L. K., Goebel, R. R. and Cercone, N. J. Structure and Organization of a Semantic Net. In N. V. Findler (ed.), *Associative Networks: Representation and Use of Knowledge by Computers*. New York: Academic Press, 1979.
 - (24) Shapiro, S. C. A net structure for semantic information storage, deduction, and retrieval. In the *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, 1971(a).
 - (25) Shapiro, S. C. The MIND System: a data structure for semantic information processing. R-837-PR, The Rand Corporation, Santa Monica, CA, August, 1971(b).
 - (26) Shapiro, S. C. The SNePS semantic network processing system. In N. V. Findler (ed.), *Associative Networks: Representation and Use of Knowledge by Computers*. New York: Academic Press, 1979.
 - (27) Simmons, R. F. Semantic Networks: Their Computation and Uses for Understanding English Sentences. In Schank, R. C. & Colby, K. M. (eds.), *Computer Models of Thought and Language*, W. H. Freeman and Company, San Francisco, 1973.
 - (28) Wilensky, Robert. Knowledge Representation – A Critique and a Proposal. In *Experience, Memory, and Reasoning*, Kolodner, J. L., and Riesbeck, C. K. (eds.). Lawrence Erlbaum Associates. Hillsdale, New Jersey. 1986.
 - (29) Woods, William A. What's in a Link: Foundations for Semantic Networks. In *Representation and Understanding: Studies in Cognitive Science*. D. G. Bobrow and A. Collins (eds.). New York: Academic Press, 1975.

SKEPTICAL REASON MAINTENANCE IS TRACTABLE

Cees Witteveen
 Delft University of Technology
 Dept of Mathematics and Computer Science
 Julianalaan 132 , 2628 BL Delft, The Netherlands
 witt@utiaab.tudelft.nl

Abstract

We introduce the three-valued skeptical semantics for nonmonotonic truth maintenance. This semantics is based on an extension of the well-known two-valued stable model semantics. Unlike the latter, however, the skeptical semantics is canonical and tractable: using a reduction to negation-free truth maintenance, the unique skeptical model can be computed in $O(n^2)$ time.

The skeptical semantics can also be used to give a better account of the belief revision problem in truth maintenance. A recent logical reconstruction of Dependency-Directed Backtracking (DDB) offers the possibility to represent different DDB-strategies by different extensions of a truth maintenance system. Given a truth maintenance system \mathcal{D} , we can distinguish a class of extensions, representing all possible DDB strategies for \mathcal{D} . Within this class there exists a unique extension whose skeptical model can be used as a canonical, information minimal belief revision model. This *skeptical belief revision* model has some important advantages:

1. the arbitrariness of solutions found by classical dependency directed backtracking methods can be avoided.
2. the semantics guarantees a (tractable) incremental updating method satisfying the weak rationality postulates of Brewka.
3. the skeptical belief revision model is easy to compute, having a worst case complexity $O(n^3)$.

1 INTRODUCTION AND MOTIVATION

The ability to maintain a coherent and consistent set of beliefs and to revise them when contradictions oc-

cur, is a major characteristic of intelligent reasoning systems.

One of the contexts in which the maintenance and revision of beliefs has been studied extensively, is the field of Reason Maintenance Systems (RMSs) (e.g. [1, 3, 4, 10, 13, 15]). In this paper, we restrict our attention to the Doyle-style or *justification-based* RMS. Basically, an RMS is a supporting system of a general reasoning system. The reasoning system communicates inferences to the RMS in the form of propositional arguments, called (*external*) *justifications*, for statements, also called *beliefs*. The RMS maintains a *dependency network* of such justifications connecting propositions. The primary tasks of an RMS are :

1. *basic reason maintenance*: to provide for a coherent and consistent interpretation of the beliefs in the network and to update the belief status after the addition of new justifications.

To capture the idea of a set of coherent, noncircular and consistent set of beliefs, the *grounded* or *stable* model semantics [6, 7] has been proposed. This semantics, however, has some clear disadvantages:

- it is *not canonical*: it is well-known [6, 7, 15] that an RMS network may have more than one stable model or even no stable model at all.
- it is *not tractable*: finding a stable model is NP-hard [6]. Hence, an RMS using this semantics is not likely to be of practical interest.

2. *belief revision*: to perform conflict-resolution in case the set of current beliefs is contradictory. Belief revision has been performed mainly by dependency-directed backtracking (DDB) algorithms. In general, the effect of DDB is the addition of a set of *internally generated* justifications J' to the set of existing (externally generated) justifications J . Belief revision then will result in a stable model of this extended set of justifications $J \cup J'$, if such a model exists and the inconsistency can be removed.

The problems with this form of belief revision are closely related to the problems of the stable model semantics :

- the outcome of the belief revision process is an arbitrary model: the selection of the internal justifications to be added is not completely determined and the (stable) model of the resulting extended network is not necessarily unique.
- sometimes an inconsistency can be removed just by selecting another stable model. Belief revision by adding justifications then will result in spurious belief models [6].
- the addition of internally generated justifications can exclude legitimate stable models of the set of *externally* generated justifications at a later time, thus causing superfluous belief revision or an impossibility to resolve a conflict.

To remove these disadvantages, Brewka [2] suggests some rationality postulates for belief revision, requiring that (1) changes produced by a single belief revision operation should be minimal w.r.t. the set of external justifications and furthermore that (2) every time belief revision is performed, the RMS should forget about the internally generated justifications in order to ensure minimality over a number of revisions as well. These postulates, however, do not completely exclude arbitrariness from the belief revision process. Moreover, combined with the 2-valued stable model semantics, these postulates almost guarantee intractability of the belief revision process.

In this paper we propose a solution to both the problems with reason maintenance and the problems with belief revision.

The problems with reason maintenance will be solved by proposing the *skeptical semantics* to capture the idea of a coherent and consistent interpretation of RMS networks. This semantics is based on a three-valued extension of the stable model semantics and offers a canonical model for each network: the skeptical model is unique and universally defined. Moreover, it can be constructed very efficiently ($O(n^2)$). This offers a solution to the basic reason maintenance problem. A detailed overview of the skeptical model and its derivation can be found in [17].

Secondly, we use a recent logical reconstruction of the DDB-idea [9] to develop two simple and efficient $O(n^3)$ belief revision strategies. We will show that these revision strategies both satisfy the suggested rationality postulates of Brewka [2].

This paper is organized as follows. In Section 2, we will give a short overview of notations, definitions and the stable model semantics together with the three-valued extensions of it. In Section 3 we discuss the skeptical model for basic reason maintenance. Section 4 presents a logical reconstruction of dependency directed backtracking and discusses two skeptical belief revision strategies. In Section 5, we discuss the rationality postulates of Brewka and we show that skeptical belief revision satisfies these postulates. Finally, Section 6 focuses on some complexity issues.

2 PRELIMINARIES

2.1 NOTATIONAL CONVENTIONS

By the perceived correspondence between truth-maintenance and propositional logic programming [6], we will use some notational conventions from logic programming in the sense of [12].

By a (basic) RMS dependency network we mean a tuple $\mathcal{D} = (N, J)$ where N is a finite set of propositional atoms and J a finite set of justifications, each justification j being a directed formula of the form

$$\alpha \wedge \beta \rightarrow c$$

where $\alpha = a_1 \wedge a_2 \wedge \dots \wedge a_m$ stands for a conjunction of positive literals in the body of the justification, $\beta = \sim b_1 \wedge \sim b_2 \wedge \dots \wedge \sim b_n$ for a conjunction of negative literals and c is a propositional atom. Both α and β may be empty. Such a justification j can be interpreted as : " c can be believed if every a_i in α is believed and none of the b_j in β is believed". Like [8], the negation operator \sim has to be conceived as *negation as failure* or *default negation* and has to be distinguished from the *classical negation* operator, to be denoted as \neg .

To avoid cumbersome notations, we will also use α and β to denote the set of literals occurring in the conjunction. We will use $A(j)$ to denote the antecedent $\alpha \wedge \beta$ of j and $C(j)$ to denote the consequent c of j . For a given wff ϕ , $\text{At}(\phi)$ will denote the set of atoms occurring in ϕ .

A justification $j = \alpha \wedge \beta \rightarrow c$ is called *monotonic* if β is the empty conjunction, i.e. the body of j does not contain any negative literal. In that case j can also be written as $\alpha \rightarrow c$. We say that a network $\mathcal{D} = (N, J)$ is *monotonic* if every justification is monotonic. *Non-monotonic* networks indicate the general case.

A (full) RMS network is a tuple $\mathcal{D}_\Delta = (N, J, \Delta)$ where $\mathcal{D} = (N, J)$ is an RMS network and Δ is a set of special justifications, called *constraints*, which can be used to declare that a given conjunction ϕ of literals forms a *nogood*, i.e. that it is inconsistent to believe in all the literals of ϕ simultaneously. Constraints are represented as justifications of the form

$$\alpha \wedge \beta \rightarrow \perp$$

where, like in justifications, α is a conjunction of atoms and β a conjunction of negated atoms.

The intended meaning of such a constraint is that it is inconsistent to believe at the same time in every a_i of α and none of the b_j in β .

2.2 INTERPRETATIONS AND MODELS

The *basic reason maintenance* task of an RMS is to give a suitable interpretation of the network $\mathcal{D} = (N, J)$. A two-valued interpretation I of \mathcal{D} is an arbitrary subset $I \subseteq N$, denoting the set of nodes believed. Since an interpretation I can be conceived as a simple truth-assignment to propositional atoms, we will use $I(a) = t$ (or $I \models a$) iff $a \in I$. Furthermore, $I(\sim a) = t$ iff $I(a) = f$. For propositions ψ over N containing \wedge , $I \models \psi$ is defined in a standard recursive way.

Note that if ψ is the empty conjunction, $I \models \psi$ for every interpretation I . An interpretation I is said to *satisfy* a justification $j = \alpha \wedge \beta \rightarrow c$, abbreviated $I \models j$, if $I \models \alpha \wedge \beta$ implies $I \models c$. An interpretation I is said to be a *model* of \mathcal{D} if $I \models j$, for every justification $j \in J$. Finally, I is said to satisfy a constraint $\alpha \wedge \beta \rightarrow \perp$ iff $I(\alpha \wedge \beta) = f$.

In case of monotonic networks, where each justification is of the form $j = \alpha \rightarrow c$, the intended semantics of the network $\mathcal{D} = (N, J)$ is given by the *minimal model* of \mathcal{D} . As a standard result (see [12]), we mention that this model equals the intersection of all propositional models of \mathcal{D} and also can be characterized as the least fixpoint $lfp(T)$ of the *immediate consequence* operator $T : 2^N \rightarrow 2^N$:

$$T(I) = \{c \mid \alpha \rightarrow c \in J, I \models \alpha\}$$

For nonmonotonic reason maintenance the (*2-valued*) *stable model* semantics offers a natural generalization of the minimal model semantics:

Definition 2.1 (Gelfond 88, Elkan 90) Let $\mathcal{D} = (N, J)$ be a network. Then M is a 2-valued stable model of \mathcal{D} iff M is the unique minimal model of the derived network $\mathcal{D}' = (N, J(M))$ where $J(M) = \{\alpha \rightarrow c \mid \alpha \wedge \beta \rightarrow c \in J, M \models \beta\}$. ■

It can be proven that M is a 2-valued stable model of \mathcal{D} iff, for every $c \in M$, a valid and well-founded argument for c can be given. That means, for every such a c , there is a sequence (c_1, c_2, \dots, c_n) such that $c_i \in M$ for $i = 1, 2, \dots, n$, $c_n = c$ and for every c_i there exists a justification $\alpha_i \wedge \beta_i \rightarrow c_i$ in J such that $M(\alpha_i \wedge \beta_i) = t$ and $\alpha_i \subseteq \{c_1, c_2, \dots, c_{i-1}\}$. Therefore, 2-valued stable models also have been called *well-founded* or *grounded* models.

Although the stable model semantics is an elegant generalization of the minimal model semantics for monotonic systems, we have some objections against this semantics.

Intuitively, a satisfactory semantics for nonmonotonic systems should at least be *universal* in the sense that it would give a suitable meaning to every network and *unique* in the sense that not more than one meaning is given to a network. Last but not least, we should keep in mind that the idea of reason maintenance is to render assistance to a reasoning system in an effective way. This means that the interpretation process should be efficient. Hence, we would require the semantics to be *tractable*.

The stable model semantics does not satisfy these requirements, since it is well-known that a network may have one, more than one, or no stable model at all. For example,

$$\mathcal{D}_1 = (\{a, b\}, \{\sim b \rightarrow a\})$$

has a unique stable model $M = \{a\}$, while

$$\mathcal{D}_2 = (\{a\}, \{\sim a \rightarrow a\})$$

does not have a stable model and

$$\mathcal{D}_3 = (\{a, b\}, \{\sim b \rightarrow a, \sim a \rightarrow b\})$$

has two stable models $M_1 = \{a\}$ and $M_2 = \{b\}$.

Also, finding a stable model has been proven to be NP-hard, so there is little hope for a tractable model construction process.

The solution to these problems can be found if we are prepared to give up the idea of a *complete* evaluation of beliefs. Instead of it, we propose a partial semantics based on three-valued logic.

2.3 THREE-VALUED INTERPRETATIONS AND MODELS

A *three-valued interpretation* I of \mathcal{D} is a tuple $I = (I_t, I_f)$ where I_t and I_f are disjoint subsets of N . I_t is called the true-set of I , I_f the false-set. The *kernel* of I , $Ker(I)$, is the set $I_t \cup I_f$ and the set of unknowns is the set $I_u = N - Ker(I)$. I as a truth-assignment $N \rightarrow \{u, f, t\}$ is defined as: $I(a) = t$ iff $a \in I_t$, $I(a) = f$ iff $a \in I_f$ and $I(a) = u$, else. The *empty* interpretation (\emptyset, \emptyset) will be denoted as \emptyset^2 .

We will use two partial orderings for the set $\{u, f, t\}$ of truth-values. The *truth ordering* $<_t$ defined by

$$f <_t u <_t t$$

is used to evaluate the truth-value of wffs. The connective \wedge is interpreted as the finite meet under the $<_t$ ordering and \sim is defined by $\sim t = f$, $\sim f = t$ and $\sim u = u$. This gives us for these operations the interpretation given in Kleene's strong three-valued logic [11, 16].

We will also use $<_t$ to define the satisfaction relation for justifications: an interpretation I is said to *satisfy* a justification $\alpha \wedge \beta \rightarrow c \in J$ iff $I(\alpha \wedge \beta) \leq_t I(c)$ ¹,

¹This is equivalent to the Lukasiewicz interpretation of the conditional.

where \leq_t is the reflexive closure of $<_t$.

Furthermore, I satisfies a constraint $\alpha \wedge \beta \rightarrow \perp \in \Delta$ iff $I(\alpha \wedge \beta) <_t t$.

I is said to be a *three-valued model* of $\mathcal{D} = (N, J)$ if I satisfies every $j \in J$. Analogously, I is said to be a *three-valued model* of the RMS \mathcal{D}_Δ iff I is a model of Δ and I satisfies every constraint in Δ . In the sequel, we will use "model" to stand for "three-valued model".

Besides a truth-ordering, we also use the *knowledge ordering* $<_k$:

$$u <_k f, u <_k t$$

This ordering will be used to compare partial interpretations and models by extending it to an ordering \sqsubseteq_k of partial interpretations defining

$$I \sqsubseteq_k I' \text{ iff } I(a) \leq_k I'(a) \text{ for every } a \in N$$

where \leq_k is the reflexive closure of $<_k$. We say that I' is the (*knowledge-*)*extension* of I . Note, that $I \sqsubseteq_k I'$ implies that $\text{Ker}(I) \subseteq \text{Ker}(I')$.

Given a set \mathcal{I} of partial interpretations, a \sqsubseteq_k -minimal interpretation is an interpretation $I \in \mathcal{I}$ such that for every $I' \in \mathcal{I}$, $I' \sqsubseteq_k I$ implies $I \sqsubseteq_k I'$.

We will need the following operations on partial interpretations :

Definition 2.2 The intersection \sqcap_k of two partial interpretations I and I' , denoted as $I \sqcap_k I'$, is the partial interpretation defined by

$$(I \sqcap_k I')(a) = I(a) \wedge_k I'(a) \text{ for all } a \in N$$

where \wedge_k is the meet under the $<_k$ ordering. ■

Partial interpretations I and I' such that $I \sqcap_k I' = \emptyset^2$ are called *disjunct* interpretations.

Definition 2.3 If I and I' are disjunct partial interpretations, the union of I and I' , denoted as $I \sqcup_k I'$, is defined as

$$(I \sqcup_k I')(a) = I(a) \vee_k I'(a) \text{ for all } a \in N$$

where \vee_k is the join under the $<_k$ ordering. ■

It is easy to see that both operations are well-defined and that the following relations do hold : $I \sqsubseteq_k (I \sqcap_k I')$ and $(I' \sqcap_k I) \sqsubseteq_k I'$.

3 THE SKEPTICAL MODEL

In this section we show first that there exists a natural generalization of the concept of a 2-valued stable model to a 3-valued stable model.

Secondly, we show that this generalization gives us back the advantages the semantics of monotonic RMS systems have: for every RMS \mathcal{D} there exists a unique \sqsubseteq_k -minimal stable model of \mathcal{D} .

Finally, we will analyze the complexity of computing such a model and show that the \sqsubseteq_k -minimal model

can be obtained in quadratic time.

For proofs of the results stated here and full details of construction we refer to [17, 18].

To introduce the notion of a 3-valued stable model, note that in the definition of a 2-valued stable model M we used a set of reduced justifications $J(M)$. In case $M = (M_t, M_f)$ is a partial interpretation for $\mathcal{D} = (N, J)$, we use two reductions of J :

1. $J(M, t) = \{\alpha \rightarrow c \mid \alpha \wedge \beta \rightarrow c \in J, M(\beta) = t\}$
2. $J(M, u) = \{\alpha \rightarrow c \mid \alpha \wedge \beta \rightarrow c \in J, M(\beta) \geq_t u\}$

Now we can define a 3-valued stable model as follows:

Definition 3.1 $M = (M_t, M_f)$ is a 3-valued stable model of $\mathcal{D} = (N, J)$ iff

1. M_t is the least 2-valued model of $\mathcal{D}(M, t) = (N, J(M, t))$
2. $M_t \cup M_u$ is the least 2-valued model of $\mathcal{D}(M, u) = (N, J(M, u))$ ■

Clearly, this definition reduces to Definition 2.1 if the partial interpretation is *complete*: in that case $\text{Ker}(M) = N$, and case 1 and case 2 of the definition above will coincide².

The following results can be obtained:

Proposition 3.2 Every RMS network \mathcal{D} has at least one 3-valued stable model.

Proposition 3.3 The intersection of all 3-valued stable models of \mathcal{D} is a 3-valued stable model of \mathcal{D} .

Note that Proposition 3.3 implies that the \sqsubseteq_k -minimal stable model of \mathcal{D} is a unique 3-valued stable model.

Example 3.4 Let $\mathcal{D} = (N, J)$, $N = \{a, b\}$ and $J = \{\sim a \rightarrow b, \sim b \rightarrow a\}$. \mathcal{D} has three stable models:

$$\begin{aligned} M_1 &= (\{a\}, \{b\}) \\ M_2 &= (\{b\}, \{a\}) \\ M_3 &= (\emptyset, \emptyset) \end{aligned}$$

The \sqsubseteq_k -minimal stable model is M_3 and M_3 is also equal to the \sqcap_k -intersection of all stable models of \mathcal{D} .

Remark. Note that the \sqsubseteq_k -minimal stable model does not always correspond to the also uniquely defined \sqsubseteq_k -minimal model of a network.

For example $\mathcal{D} = (\{a\}, \{a \rightarrow a\})$ has a unique \sqsubseteq_k -minimal model \emptyset^2 , but the unique \sqsubseteq_k -minimal stable model is $(\emptyset, \{a\})$. ■

²The definition of 3-valued stable models appearing in [14] is completely equivalent to the definition presented here.

In the semantics for logic programming, the \sqsubseteq_k -minimal stable model has been baptized as the *well-founded model* [14]. Since this namegiving will cause a possible confusion in reason maintenance, we propose to call it the *skeptical model*.

The skeptical model offers a solution to the problems encountered in the 2-valued stable model semantics:

- **universality**

By Proposition 3.2, every network has a 3-valued stable model, hence by Proposition 3.3, the \sqsubseteq_k -minimal stable model also exists.

- **uniqueness**

The \sqsubseteq_k -minimal stable model is unique.

- **tractability**

Elsewhere ([17]) we have shown that for every network $\mathcal{D} = (N, J)$, the skeptical model can be computed in $O(|N| \times |\mathcal{D}|)$. More precisely, if M is the skeptical model of \mathcal{D} , it can be constructed in $O(|\text{Ker}(M)| \times |\mathcal{D}|)$.

Since the details of this algorithm are not essential to an understanding of the peculiarities of the model, we have presented some of the construction ideas in the Appendix.

In the next section, we will use the skeptical model to offer a solution to the belief revision problem.

Remark. The skeptical model M_{sk} of an RMS \mathcal{D} can be interpreted as an information minimizing model. As an interesting property for reason maintenance it can be shown that not only the set of all its true beliefs is contained in the true-belief set of every stable model of \mathcal{D} , but also that every grounded argument for such a belief valid in M is a grounded argument valid in every other stable model of \mathcal{D} . ■

4 SKEPTICAL BELIEF REVISION

Belief revision has to be performed if the set of current beliefs does not satisfy all the constraints in Δ . The 2-valued stable model semantics has some annoying properties in this respect. As Elkan [6] remarks, this semantics can induce spurious belief revision, since the choice for one particular stable model could lead to the need for belief revision (by DDB, for example), although it is possible that the constraints could be satisfied just by choosing another stable model for the original system.

Of course, to avoid such spurious belief revision, one could search for all possible stable models of the original network. Unfortunately, however, the problem to find all stable models is intractable as the number of

stable models can be exponential in the number of atoms.

Using a skeptical model semantics, this problem does not arise: our first result concerns a tell-tale property the skeptical model M_{sk} of a basic RMS $\mathcal{D} = (N, J)$ has: if M_{sk} does not satisfy the constraints Δ of $\mathcal{D}_\Delta = (N, J, \Delta)$, no other (2- or 3-valued) stable model of \mathcal{D} will do. This means that, whatever method for belief revision we will actually use, spurious belief revision can be excluded using the skeptical model semantics.

Proposition 4.1 *If M_{sk} is the skeptical model of \mathcal{D} but not a model of \mathcal{D}_Δ , then \mathcal{D}_Δ has no 2- or 3-valued stable model at all.*

PROOF. Let M_{sk} be the skeptical model of \mathcal{D} and suppose M_{sk} is not a model of \mathcal{D}_Δ . Assume, on the contrary, that there exists a stable model $M' \neq M_{sk}$ of \mathcal{D}_Δ .

Since M_{sk} is the \sqsubseteq_k -minimal model of \mathcal{D} and M' is a stable model of \mathcal{D} , $M_{sk} \sqsubseteq_k M'$. Since M_{sk} does not satisfy every constraint in Δ , there is a $j \in \Delta$ such that $M_{sk}(A(j)) = t$. Hence, since $A(j)$ is a conjunction of literals,

$$M_{sk}(x) = t$$

for every literal $x \in A(j)$.

However, since M' is a model of \mathcal{D}_Δ , $M'(A(j)) <_t t$. Hence, there exists a literal x_0 in $A(j)$ such that

$$M'(x_0) <_t t$$

Since $M_{sk} \sqsubseteq_k M'$, this implies that

$$M_{sk}(x_0) \in \{u, f\}$$

and therefore,

$$M_{sk}(A(j)) <_t t$$

for every such a $j \in \Delta$. But then M_{sk} is also a stable model of \mathcal{D}_Δ ; contradiction. Therefore, such an M' cannot exist. □

Note that Proposition 4.1 is also useful in its contrapositive form: If there exists a stable model of \mathcal{D}_Δ , the skeptical model M_{sk} of \mathcal{D} is also a model of \mathcal{D}_Δ . Hence, whatever other stable model we may have in mind for \mathcal{D}_Δ , the skeptical model could have been chosen as well.

Now suppose M_{sk} is not a model of \mathcal{D}_Δ . In that case, Proposition 4.1 shows that the only possibility is to adapt the network itself in order to avoid inconsistencies.

In the next section we will discuss DDB as such an adaptation technique, first showing that it can be reconstructed in a logical way and secondly, that the skeptical model has some interesting properties with respect to this reconstruction.

4.1 A LOGICAL CHARACTERIZATION OF DEPENDENCY DIRECTED BACKTRACKING

As we already remarked in the introduction, belief revision has been performed mainly by dependency-directed backtracking (DDB), the effect of which can be described as the addition of a set of *internally generated* justifications J' to the set of existing (externally generated) justifications J .

A stable model for this extended set of justifications, satisfying the set of constraints, will count as a solution to the belief revision problem.

What is the rationale behind such a belief revision method? Looking back at the idea of a stable model, we see that it is closely connected to the use of the Closed World Assumption (CWA) : unless you have evidence for believing p , do not believe it. In a basic RMS, the application of the CWA is harmless. If we add constraints, however, the application of the CWA may lead to undesirable consequences such as belief in absurdities.

In order to avoid such contradictory beliefs, we need to repair the effects of applying the CWA by selective application of the dual notion of the *Open World Assumption* : unless you have good reasons for believing *not* p , you are allowed to believe p . As we will see, such a selective application of the open world assumption can be implemented by the addition of justifications motivated by the occurrence of a contradiction.

The logical reconstruction of DDB given by Giordano & Martelli [9] in fact can be viewed as an attempt to implement the open world assumption by adding *contrapositives* of justifications to a given set J . In this paper they argued that, logically, the application of DDB comes down to the addition of *contrapositives* of justifications to J . For example, a justification $a \wedge b \rightarrow \perp$ can be seen to generate the additional arguments $a \rightarrow \neg b$ and $b \rightarrow \neg a$ in order to get rid of the contradiction. Adding such a set of contrapositives or *backward justifications* $B(J)$ to J may result in an extended system, having a model that satisfies all the constraints.

Unfortunately, Giordano and Martelli based their analysis on the 2-valued stable model semantics. We will adapt and extend their idea to the three-valued case.

Following [9], the set $B(j)$ of backward justifications for a given justification $j = \alpha \wedge \beta \rightarrow c$ can be defined as

$$B(j) = B(j, \alpha) \cup B(j, \beta)$$

where

$$B(j, \alpha) = \{\alpha' \wedge \beta \wedge \neg c \rightarrow \neg a \mid \alpha' = \alpha - \{a\}, a \in \alpha\}$$

and

$$B(j, \beta) = \{\alpha \wedge \beta' \wedge \neg c \rightarrow b \mid \beta' = \beta - \{\sim b\}, b \in \beta\}$$

In these backward justifications the classical negation operator \neg will be used. To apply the stable model

machinery, these extended forms of justifications will be *compiled* into a standard RMS by treating negated atoms $\neg a$ as new positive atoms and we will add some justifications and constraints to the RMS to preserve the meaning of such special atoms (cf [8]).

Remark. Strictly speaking, it is not necessary to create backward justifications for all justifications $j \in J$. It suffices, for example, to take those justifications whose (in)direct consequence is \perp . Therefore, let us define the set of *direct ancestors* $Anc(c)$ of an element $c \in N$ as

$$Anc(c) = \{a \mid \exists j \in J [a \in A(j) \wedge C(j) = c]\}$$

and let $Anc^*(c)$ denote the transitive closure of $Anc(c)$. Given a network $D_\Delta = (N, J, \Delta)$, as the set of backward justifications we can take the set

$$B(J) = \bigcup_{j \in J \cup \Delta, C(j) \in Anc^*(\perp)} B(j)$$

In the sequel, we will assume that the set of backward justifications has been determined relative to $Anc^*(\perp)$. Of course, more refined definitions of $B(J)$ are possible and will be developed. ■

To preserve the meaning of $\neg x$ as the "classical" negation of x , we have to add, like was done implicitly in [9], for every such a $\neg x$ a constraint

$$\delta_x \equiv x \wedge \neg x \rightarrow \perp$$

to Δ in order to ensure that not both x and $\neg x$ can be true.

But we also have to add, for every such a $\neg x$, a justification

$$j_{\neg x} \equiv \neg \neg x \rightarrow x$$

in order to ensure that not both x and $\neg x$ can be false. Note that this justification is an almost direct implementation of the Open World Assumption in a dependency network.

Let $B(J)$ be the union of all sets $B(j)$, $j \in J \cup \Delta$, let $\neg N$ be a set of atoms denoting the set of classically negated atoms occurring in $B(J)$, and let $J_{\neg N}$ be the set of all justifications $j_{\neg x}$, $\neg x \in \neg N$.

Then, given a network $D_\Delta = (N, J, \Delta)$, we can define the *full DDB-transform* $B_2(D_\Delta)$ of D_Δ as

$$B_2(D_\Delta) = (N_2, J_2, \Delta_2)$$

where

$$N_2 = N \cup \neg N$$

$$J_2 = J \cup B(J) \cup J_{\neg N}$$

$$\Delta_2 = \Delta \cup \{\delta_x \mid \neg x \in \neg N\}$$

In the literature, several varieties of DDB are described, each using a special search strategy to identify atoms to be justified by internally generated justifications.

To take these varieties into account, we will consider a class of DDB transforms of a given network. Therefore, we will define the class $\mathcal{B}_2(\mathcal{D}_\Delta)$ as the set of all (sub)networks

$$b_2(\mathcal{D}_\Delta) = (N', J', \Delta')$$

of the full DDB transform $B_2(\mathcal{D}_\Delta) = (N_2, J_2, \Delta_2)$, where

1. $J \subseteq J' \subseteq J_2$ contains a subset of the backward justifications $B(J)$ of J_2 .
2. N' and Δ' are corresponding subsets of respectively N_2 and Δ_2 .

Every such a network $b_2(\mathcal{D}_\Delta)$ represents the application of a particular DDB strategy.

Definition 4.2 M is a belief revision model of \mathcal{D}_Δ if M is a stable model of some $b_2(\mathcal{D}_\Delta) \in \mathcal{B}_2(\mathcal{D}_\Delta)$.

Note that even if we fix the ddb-strategy, such a belief revision model M does not need to be unique, since $b_2(\mathcal{D}_\Delta)$ may have more than one stable model.

Now we could obtain a *canonical* model for each *particular* ddb-strategy by defining the skeptical model of $b_2(\mathcal{D}_\Delta)$ to be the belief revision model for that strategy.

A more general approach, however, is preferred. We will show that the full DDB transform $B_2(\mathcal{D}_\Delta)$ in a certain sense is a canonical DDB-strategy within the class $\mathcal{B}_2(\mathcal{D}_\Delta)$ of strategies, since its skeptical model (if it exists) can be used as a canonical belief revision model.

We will call this skeptical model the *strong* belief revision model of \mathcal{D}_Δ , since its canonicity does hold with respect to the class of 2-valued stable models of ddb systems in $\mathcal{B}_2(\mathcal{D}_\Delta)$. Moreover, this belief revision model will be a 2-valued stable model, if there is a unique 2-valued stable solution to the backtracking problem.

Definition 4.3 The strong (skeptical) belief revision model of \mathcal{D}_Δ , denoted as M_{B_2} , is the skeptical model of the full DDB-transform $B_2(\mathcal{D}_\Delta)$.

We claim that this model is canonical in the following sense:

Theorem 4.4 M_{B_2} is an indicative model in the sense that if $B_2(\mathcal{D}_\Delta)$ does not have a skeptical model, no $b_2(\mathcal{D}_\Delta) \in \mathcal{B}_2(\mathcal{D}_\Delta)$ has a belief revision model, i.e. there are solutions to the belief revision problem only if M_{B_2} exists.

Theorem 4.5 M_{B_2} is a minimal belief revision model in the sense that for every 2-valued belief revision model M of some $b_2(\mathcal{D}_\Delta)$, $M_{B_2}(n) \leq_k M(n)$, for every $n \in N$.

Before we prove these theorems, a simple lemma is needed to show that every 2-valued model M of any $b_2(\mathcal{D}_\Delta) \in \mathcal{B}_2(\mathcal{D}_\Delta)$ preserves the intended meaning of $\neg x$ as the classical negation of $x \in N$. Here, as usual, \neg is defined by $\neg t = f$ and $\neg f = t$.

Lemma 4.6 Let $\mathcal{D} = (N, J)$ and let M be a 2-valued model of some $b_2(\mathcal{D}_\Delta) = (N', J', \Delta')$. Then, for every $\neg x \in N'$, $M(\neg x) = \neg M(x)$.

PROOF. If $M(\neg x) = f$, $M(\sim \neg x) = t$, hence $M(x) = t$, since $\sim \neg x \rightarrow x \in J'$ and M is a model. If $M(\neg x) = t$, $M(x) \neq t$, since M satisfies the constraint $x \wedge \neg x \rightarrow \perp$. Hence, since M is 2-valued, $M(x) = f$. \square

PROOF OF THEOREM 4.4. Let $\mathcal{D}_\Delta = (N, J, \Delta)$ and let M_2 be a 2-valued stable model of some $b_2(\mathcal{D}_\Delta) = (N_2, J_2, \Delta_2)$. We construct a 2-valued interpretation $M'_2 \subseteq N'_2$ for the full DDB-transform

$$B_2(\mathcal{D}_\Delta) = (N'_2, J'_2, \Delta'_2)$$

as follows:

$$M'_2(x) = \begin{cases} M_2(x) & \text{if } x \in N_2 \\ \neg M_2(y) & \text{if } x = \neg y \text{ and } \neg y \in N'_2 - N_2 \end{cases}$$

So M'_2 is a simple extension of M_2 to N'_2 . Note that M'_2 is a two-valued interpretation of $B_2(\mathcal{D}_\Delta)$.

We will prove that M'_2 is a stable model of $B_2(\mathcal{D}_\Delta)$. Then, by Proposition 4.1, it follows that M_{B_2} exists.

By Lemma 4.6 and the construction of M'_2 , it is easy to see that M'_2 preserves the intended meaning of $\neg x$ for every $\neg x \in N'_2$.

Now, let $M''_2 = M'_2 \cap N$ let $B'_2(\mathcal{D}_\Delta)$ be equal to $B_2(\mathcal{D}_\Delta)$, except that every atom $\neg x$ is interpreted as the formula $\neg x$. Since M'_2 preserves the meaning of $\neg x$, it follows that

$$\begin{array}{c} M'_2 \text{ is a model of } B_2(\mathcal{D}_\Delta) \\ \iff \\ M''_2 \text{ is a model of } B'_2(\mathcal{D}_\Delta) \end{array}$$

By construction of M'_2 and M''_2 , we have

$$M_2 \cap N = M'_2 \cap N = M''_2$$

Since M_2 is a model of $b_2(\mathcal{D}_\Delta)$, $M_2 \cap N$ is a model of \mathcal{D}_Δ . Hence, $M_2 \cap N$ must be a model of $B'_2(\mathcal{D}_\Delta)$ as well, since $B'_2(\mathcal{D}_\Delta)$ contains only justifications in J , their logical contrapositives and 'tautologies' like $\sim \neg x \rightarrow x$.

Hence, M''_2 is a model of $B'_2(\mathcal{D}_\Delta)$ and therefore, M'_2 is a model of $B_2(\mathcal{D}_\Delta)$.

The proof that M'_2 is also a 2-valued stable model of $B_2(\mathcal{D}_\Delta)$ is easy and is left to the reader. \square

PROOF OF THEOREM 4.5 Let M_2 be a 2-valued stable model of some $b_2(\mathcal{D}_\Delta) = (N_2, J_2, \Delta_2)$. By the construction given in the proof above, there exists a stable model M'_2 of $B_2(\mathcal{D}_\Delta)$ such that $M_2(n) = M'_2(n)$

for every $n \in N$. By Proposition 4.1, M_{B_2} exists and since $M_{B_2}(n) \leq_t M'_2(n)$ for every $n \in N$, the theorem follows. \square

We present some illustrative examples.

Example 4.7 Consider the RMS \mathcal{D} with the following set of justifications J :

$$\begin{aligned} &\rightarrow \text{switch_on} \\ &\text{switch_on} \wedge \sim \text{broken} \rightarrow \text{light_on} \\ &\rightarrow \text{light_off} \end{aligned}$$

and a constraint

$$\Delta = \{ \text{light_on} \wedge \text{light_off} \rightarrow \perp \}$$

The set $B(J)$ of all backward justifications is

$$\begin{aligned} &\text{switch_on} \wedge \sim \text{light_on} \rightarrow \text{broken} \\ &\sim \text{light_on} \wedge \sim \text{broken} \rightarrow \sim \text{switch_on} \\ &\text{light_on} \rightarrow \sim \text{light_off} \\ &\text{light_off} \rightarrow \sim \text{light_on} \end{aligned}$$

The following justifications are added to preserve the meaning of the negated atoms:

$$\begin{aligned} &\sim \sim \text{switch_on} \rightarrow \text{switch_on} \\ &\sim \sim \text{light_on} \rightarrow \text{light_on} \\ &\sim \sim \text{light_off} \rightarrow \text{light_off} \end{aligned}$$

and, finally, three constraints have to be added:

$$\begin{aligned} &\text{switch_on} \wedge \sim \text{switch_on} \rightarrow \perp \\ &\text{light_on} \wedge \sim \text{light_on} \rightarrow \perp \\ &\text{light_off} \wedge \sim \text{light_off} \rightarrow \perp \end{aligned}$$

All these sets occur in $B_2(\mathcal{D}_\Delta)$.

The skeptical model is

$$M_{B_2} = (\{\text{switch_on}, \text{broken}, \text{light_off}, \sim \text{light_on}\}, \{\text{light_on}, \sim \text{switch_on}, \sim \text{light_off}\})$$

and shows nicely the result expected: *broken* is believed, because otherwise the contradiction is believed. Note that this strong belief revision model is a 2-valued stable model of $B_2(\mathcal{D})$.

Example 4.8 Consider the well-known Nixon Diamond:

$$\begin{aligned} &\text{republican} \wedge \sim \text{abrepublican} \rightarrow \text{hawk} \\ &\text{quaker} \wedge \sim \text{abquaker} \rightarrow \text{dove} \\ &\rightarrow \text{republican} \\ &\rightarrow \text{quaker} \\ &\text{hawk} \wedge \text{dove} \rightarrow \perp \end{aligned}$$

The skeptical model M_{sk} of this RMS evaluates republican, quaker, hawk, dove and \perp to *true* and the atoms abrepublican and abquaker to *false*.

Since $M_{sk}(\perp) = t$, we have to perform belief revision and the resulting belief revision model is:

$$M_{B_2} = (\{\text{republican}, \text{quaker}\}, \emptyset)$$

In this case, the addition of contrapositives results in evaluating both "ab" atoms to *unknown*, which, intuitively is more acceptable than an arbitrary choice for either making abrepublican true or abquaker true.

Example 4.9 We will show that although every 2-valued model preserves the meaning of the atoms $\neg a$, the skeptical model does not necessarily do so:

let $\mathcal{D}_\Delta = (N, J, \Delta)$ where $N = \{a, b, c\}$ $J = \{\sim b \wedge \sim c \rightarrow a\}$ and $\Delta = \{a \rightarrow \perp\}$. \mathcal{D}_Δ has no skeptical model, so belief revision is necessary.

Now $B_2(\mathcal{D}_\Delta) = (N', J', \Delta)$, where

$$\begin{aligned} N' &= N \cup \{\neg a\} \\ J' &= J \cup \{\neg a \wedge \sim b \rightarrow c, \neg a \wedge \sim c \rightarrow b\} \\ &\quad \cup \{\rightarrow \neg a, \sim \neg a \rightarrow a\} \\ \Delta' &= \Delta \cup \{\neg a \wedge a \rightarrow \perp\} \end{aligned}$$

and $M_{B_2} = (\{\neg a\}, \emptyset)$. But then $M_{B_2}(a) = u$ while $M_{B_2}(\neg a) = t$.

4.2 WEAK SKEPTICAL BELIEF REVISION

The "classical" negation operator \neg for three-valued logic is defined by $\neg t = f, \neg f = t$ and $\neg u = u$. The last example shows some defects in the skeptical belief revision model M_{B_2} : the meaning of the atoms $\neg a$ as the three-valued negation of a is not always preserved. This defect, however, can be easily removed, if we make some change in the form of the backward justifications $B(j)$.

We will change the backward justifications as follows:

Definition 4.10 Let $j = \alpha \wedge \beta \rightarrow c$ be a justification. The set $B'(j)$ of all modified backward justifications of j is defined as

$$B'(j) = B'(j, \alpha) \cup B'(j, \beta)$$

where

$$B'(j, \alpha) = \{\alpha' \wedge \beta' \rightarrow \neg a \mid \alpha' = (\alpha - \{a\}) \wedge \neg c, \beta' = (\beta - \{\sim b\}) \wedge \sim \neg b, b \in \beta\}$$

and

$$B'(j, \beta) = \{\alpha' \wedge \beta' \rightarrow b \mid \alpha' = \alpha \wedge \neg c, \beta' = (\beta - \{\sim b\}) \wedge \sim \neg b, b \in \beta\}$$

For this modified weak belief revision form, we define the full DDB-transform $B_3(\mathcal{D}_\Delta)$ and the class $\mathcal{B}_3(\mathcal{D}_\Delta)$ in the same way as we did for the original DDB transform $B(J)$.

First of all, it is not difficult to see that with respect to

the set of 2-valued stable models, $\mathcal{B}_2(\mathcal{D}_\Delta)$ and $\mathcal{B}_3(\mathcal{D}_\Delta)$ do not differ.

Let

$$\mathcal{M}_y^x(\mathcal{D}_\Delta) = \{M \mid M \text{ is an } x\text{-valued stable model of some } b_y(\mathcal{D}_\Delta) \in \mathcal{B}_y(\mathcal{D}_\Delta)\}$$

where $x, y \in \{2, 3\}$.

Then we have the following result:

Proposition 4.11 *Let \mathcal{D}_Δ be a dependency network. Then*

$$\mathcal{M}_2^2(\mathcal{D}_\Delta) = \mathcal{M}_3^2(\mathcal{D}_\Delta)$$

PROOF. Left to the reader. \square

Analogously to Lemma 4.6, we can show that for weak belief revision, every 3-valued stable model of $b_3(\mathcal{D}_\Delta) = (N', J', \Delta')$ in $\mathcal{B}_3(\mathcal{D}_\Delta)$ preserves the meaning of $\neg x \in N'$.

Lemma 4.12 *For every $M \in \mathcal{M}_3^3(\mathcal{D}_\Delta)$, if M is a 3-valued stable model of $b_3(\mathcal{D}_\Delta) = (N', J', \Delta')$, $M(\neg x) = \neg M(x)$ for every $\neg x \in N'$.*

PROOF We distinguish the following cases :

1. $M(\neg x) = f$.

So $M(\sim \neg x) = t$ and therefore, since $\sim \neg x \rightarrow x \in J'$ and M is a model of \mathcal{D} , $M(x) = t$.

2. $M(\neg x) = u$.

Since $M(\sim \neg x) = u$, $\sim \neg x \rightarrow x \in J'$ implies that $M(x) \geq_t u$.

Now every justification j for $\neg x$ is of the form

$$j = (\alpha - \{x\}) \wedge \neg c \wedge \beta \wedge \sim x \rightarrow \neg x$$

and, since M is a minimal model, $M(\neg x) = u$ implies that $M(\sim x) \geq_t u$. Hence, $M(x) \leq_t u$, and therefore, $M(x) = u$.

3. $M(\neg x) = t$.

Since every justification j for $\neg x$ is of the form

$$j = (\alpha - \{x\}) \wedge \neg c \wedge \beta \wedge \sim x \rightarrow \neg x$$

$M(\sim x) = t$, so $M(x) = f$.

So, in every case, $M(\neg x) = \neg M(x)$. \square

Note that this construction makes the constraints $x \wedge \neg x \rightarrow \perp$ obsolete, since $M(\neg x) = t$ implies $M(x) = f$ for a stable model M , without using this constraint. Let $b'_3(\mathcal{D}_\Delta)$ be equal to $b_3(\mathcal{D}_\Delta) = (N', J', \Delta')$ except that all constraints $x \wedge \neg x \rightarrow \perp$ have been removed from Δ' . Then the following corollary is immediate:

Corollary 4.13 *Let \mathcal{D}_Δ be an arbitrary RMS. Then, for every $b_3(\mathcal{D}_\Delta) \in \mathcal{B}_3^3(\mathcal{D}_\Delta)$, M is a 3-valued stable model of $b_3(\mathcal{D}_\Delta)$ iff M is a 3-valued stable model of $b'_3(\mathcal{D}_\Delta)$.*

The following theorem is a main result of this paper. It shows that the belief revision model M_{B_3} is a straightforward generalization of the skeptical model semantics and the 2-valued minimal model semantics: while the skeptical model semantics is a \sqsubseteq_k -minimal stable model for a given network \mathcal{D}_Δ , the 3-valued weak belief revision model M_{B_3} is the \sqsubseteq_k -minimal stable model for the class of networks $\mathcal{B}_3(\mathcal{D}_\Delta)$:

Theorem 4.14 *Let M be a 3-valued stable model for some $b_3(\mathcal{D}_\Delta) = (N', J', \Delta')$. Let M_{B_3} be the skeptical model of $B_3(\mathcal{D}_\Delta)$. Then, for every $a \in N'$, $M_{B_3}(a) \leq_k M(a)$.*

PROOF Let the full DDB-transform $B_3(\mathcal{D}_\Delta) = (N_3, J_3, \Delta_3)$. Consider the following interpretation M' for $B_3(\mathcal{D}_\Delta)$:

$$M'(x) = \begin{cases} M(x) & \text{if } x \in N' \\ \neg M(y) & \text{if } x = \neg y \in N_3 - N' \end{cases}$$

Clearly, M' is a model of $B_3(\mathcal{D}_\Delta)$. For, let $x \in N_3$ and suppose on the contrary that there exists a justification $j \in J_3$ such that

$$C(j) = x \text{ and } M'(x) <_t M'(A(j))$$

Since M is a model of $b_3(\mathcal{D}_\Delta)$, M is also a model of \mathcal{D}_Δ . Therefore, $j \notin J \cup \Delta$. Hence,

$$At(j) \cap \neg N \neq \emptyset$$

We distinguish two cases.

1. If $x \in \neg N$, $x = \neg y$ for some $y \in N$ and $A(j)$ must contain an occurrence of $\sim y$. Hence,

$$M'(A(j)) \leq_t M'(\neg y)$$

and by Lemma 4.12 (applied to M),

$$M'(A(j)) \leq_t \sim M'(y) = \sim \neg M(\neg y) = M(\neg y)$$

But then

$$M'(A(j)) \leq_t \neg M(y) = M'(x)$$

contradicting the assumption that M' does not satisfy j .

2. If $x \in N$, $A(j)$ must contain some atom $\neg y \in \neg N$ and j is a backward justification of some $j_0 \in J \cup \Delta$.

Since j must be of the form

$$\alpha \wedge \neg y \wedge (\beta - \{\sim x\}) \wedge \sim \neg x \rightarrow x$$

By construction of M' , $M'(\neg x) = \neg M(x)$. But then it follows immediately that $M'(\sim \neg x) = \sim \neg M(x) = M(x)$. Since $x \in N$, $M'(x) = M(x)$. Hence, $M'(At(j)) \leq_t M'(\sim \neg x) \leq_t M'(x)$, contradicting the assumption that M' does not satisfy j .

So in every case, $M'(j) = t$, contradicting our assumption. Hence, M' is a model of $B_3(\mathcal{D}_\Delta)$. The proof that M' is a stable model is easy and left to the reader.

Since M' exists, by Proposition 4.1, M_{B_3} must exist. Since $M_{B_3} \sqsubseteq_k M'$ for every stable model M' of $B_3(\mathcal{D}_\Delta)$, it follows that $M_{B_3}(a) \leq_k M(a)$ for every stable model M of a ddb network $b_3(\mathcal{D}_\Delta = (N', J', \Delta'))$ and every $a \in N'$. \square

5 SKEPTICAL BELIEF REVISION AND BREWKA'S RATIONALITY POSTULATES

In [1] Brewka discusses some proposals for implementing belief revision in truth maintenance.

His objection against the currently used dependency directed backtracking methods is that they violate some rather weak rationality postulates every conflict resolution method should obey.

In his view, the main cause for this failure is a lack of distinction between two classes of justifications in such systems : the *external* justifications provided by some external problem solver and the *internally generated* justifications which are the result of the conflict resolution method. Brewka shows that such a failure can result in a method that succeeds in producing minimal changes in each step but behaving inadequately viewed over a number of steps.

Slightly adapting his terminology, Brewka stated the following rationality postulates :

1. Rationality postulate 1

The changes performed by the RMS to remove inconsistencies have to be minimal with respect to a given criterion.

2. Rationality postulate 2

The belief revision model M of a DDB revision network should be t -minimal in the sense that there is no other DDB revision network having a stable model M' such that $M'_t \subseteq M_t$, i.e. M should make true a minimal number of atoms in order to remove the inconsistency.

3. Rationality postulate 3

Minimality of changes has to be determined with respect to the external information, i.e. the external justifications, available to the RMS.

Both skeptical models of belief revision can satisfy all these rationality postulates :

Theorem 5.1 *The skeptical belief revision models M_{B_2} and M_{B_3} satisfy the rationality postulates of Brewka w.r.t. to the corresponding class of DDB-strategies.*

PROOF

- To satisfy Rationality postulate 1, it is only required to state a minimality criterion. In the ddb-belief revision strategy we could choose as a minimum the number of justifications whose contrapositives are added.

With respect to the full DDB-transforms, we can choose the set of justifications whose conclusions occur in $Anc^*(\perp)$. We know that w.r.t. the corresponding classes of ddb-transforms, $B_2(\mathcal{D}_\Delta)$ and $B_3(\mathcal{D}_\Delta)$ then contain the minimal number of contrapositives added to J , since in some cases only these full ddb-transforms will have a belief revision model.

- M_{B_x} for $x = 2, 3$ satisfies rationality postulate 2, since by Theorem 4.5 and Theorem 4.14, if there exists a stable belief revision model of some $b_x(\mathcal{D}_\Delta)$ such that $M_t \subseteq (M_{B_x})_t$, this implies that $(M_{B_x})_t = M_t$, since M_{B_x} is \sqsubseteq_k -minimal.
- To satisfy the third postulate, every time a new external justification is added to \mathcal{D}_Δ , we first compute the skeptical model for the new basic RMS. If this model exists, we are done. Otherwise, since the skeptical model does not exist, we know that no other stable model will be found and therefore, DDB can be performed.

\square

6 COMPLEXITY ISSUES

As proved in [17], determining the skeptical model for \mathcal{D} can be done in $O(|N| \times |\mathcal{D}|)$. Here, $|\mathcal{D}|$ stands for the size of \mathcal{D} which is equal to

$$|\mathcal{D}| = |N| + \sum_{j \in J} l(j)$$

where $l(j)$ is the length of j considered as a string of atoms and logical symbols, each atom being of a unit length. We will abbreviate $\sum_{j \in J} l(j)$ by $\| J \|$. Given an RMS \mathcal{D}_Δ , determining and verifying the skeptical model is bounded above by

$$O(|N| \times |\mathcal{D}_\Delta|) = O(|N|^2 + |N| \times \| J \cup \Delta \|)$$

Determining the skeptical belief revision model means that we have to find the skeptical model $B_x(\mathcal{D}_\Delta) = (N_x, J_x, \Delta_x)$, $x = 2, 3$. Since

1. $|N_x| \leq 2 \times |N|$
2. $\| J_x \cup \Delta_x \| \leq (|N| + 1) \times \| J \cup \Delta \|$

the construction of the skeptical model M_{B_x} will cost $O(|N_x|^2 + |N_x| \times \| J_x \cup \Delta_x \|) = O(|N|^2 \times \| J \cup \Delta \|)$ assuming that $\| J \cup \Delta \| \geq 1$. Roughly, this amounts to a complexity of $O(|\mathcal{D}_\Delta|^3)$.

7 CONCLUSIONS

We have introduced the skeptical semantics for non-monotonic truth maintenance. With respect to belief revision problems, it turns out that irrespective of the revision method used, the skeptical semantics has an important advantage: spurious belief revision can be avoided in any case.

If belief revision is necessary, we can apply different revision strategies. In this paper we have concentrated on the use of dependency directed backtracking strategies and we have shown that the skeptical approach can be used to obtain a canonical belief revision model. In fact we distinguished two belief revision models, a strong belief revision model and a weak belief revision model. Both models are skeptical models of a transformed reason maintenance system and are information minimal w.r.t. a class of stable belief revision models. The strong belief revision model is information minimal w.r.t. all 2-valued belief revision models occurring as the outcome of a DDB-strategy, while the weak belief revision model is the information minimal stable model with respect to all stable belief revision models we can produce using DDB.

References

- [1] Brewka, G., Nonmonotonic Reasoning - From Theoretical Foundations Towards Efficient Computation, thesis University of Hamburg, 1989.
- [2] Brewka, G., On Minimal Change: A New Architecture for Nonmonotonic TMS, Unpublished manuscript, GMD Bonn, 1990.
- [3] Charniak, E., Riesbeck C., and McDermott, D., *Artificial Intelligence Programming*, L.E. Erlbaum, Baltimore, 1979.
- [4] Doyle, J., A Truth Maintenance System, *Artificial Intelligence*, 12, 1979
- [5] Dowling, W. and Gallier, J., Linear Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae, *Journal of Logic Programming* 3 (1984), 267-284.
- [6] Elkan, Ch., Logical Characterizations of Non-monotonic TMSs, in: Kreczmar, A. and G. Mirkowska, (eds) *Mathematical Foundations of Computer Science 1989*, Springer Heidelberg, 1989, pp. 218-224.
- [7] Gelfond, M., and Lifschitz, V., The Stable Model Semantics for Logic Programming. In : *Fifth International Conference Symposium on Logic Programming*, pp. 1070-1080, 1988.
- [8] Gelfond,M., and Lifschitz, V., Logic Programs with Classical Negation. in: D.H. Warren & P. Szeredi (eds), *Proceedings of the 7th International Conference on Logic Programming*, pp. 579-597, 1990.
- [9] Giordano, L., Martelli, A., Generalized Stable Models, Truth Maintenance and Conflict Resolution, in : D.H. Warren & P. Szeredi (eds), *Proceedings of the 7th International Conference on Logic Programming*, pp. 427-441, 1990.
- [10] Goodwin, J. , An Improved Algorithm for Non-Monotonic Dependency Net Update. LITH-MAT-R-82-23, Linköping University 1982.
- [11] Kleene, S., *Introduction of metamathematics*, Van Nostrand, 1952.
- [12] Lloyd, J.W., *Foundations of Logic Programming*, Springer Verlag, Heidelberg, 1987
- [13] Petrie, C.J., Revised Dependency-Directed Backtracking for Default Reasoning, *Proceedings AAAI*, 1987.
- [14] Przymusinska, H. and Przymusinski, T., Semantic Issues in Deductive Databases and Logic Programs, in: *Formal Techniques in Artificial Intelligence, A Sourcebook*, R.B. Banerji (ed.), Elsevier, Amsterdam, 1990, pp. 321-367.
- [15] Reinfrank, M., Fundamentals and Logical Foundations of Truth Maintenance, Linköping Studies in Science and Technology. Dissertations no. 221, Linköping University, 1989.
- [16] Turner, R., *Logics for Artificial Intelligence*, Ellis Horwood Ltd, Chichester, 1987.
- [17] Witteveen, C., Fixpoint Semantics for Truth Maintenance, TWI-report 90-85, Faculty of Technical Mathematics and Computer Science, Delft University of Technology, Delft, 1990.
- [18] Witteveen, C., Partial Semantics for Truth Maintenance, to appear as TWI-Report, Department of Mathematics and Computer Science, Delft University of Technology, 1990.
- [19] Witteveen, C., Partial Semantics for Truth Maintenance, in : J.W. van Eyk(ed.), *Proceedings of JELIA90*, LNCAI, Springer Heidelberg, 1990, (to appear).

8 APPENDIX

In this section we will give some details about the construction of the skeptical model. First we need some definitions. Given a three-valued or partial interpretation I of a network $\mathcal{D} = (N, J)$, we can distinguish two subnetworks of \mathcal{D} induced by I :

- the network \mathcal{D} restricted to I , denoted by \mathcal{D}_I , containing only atoms occurring in $\text{Ker}(I)$ and only those justifications $j \in J$ with $At(j) \subseteq \text{Ker}(I)$. Note that I is a 2-valued interpretation for \mathcal{D}_I .
- the network \mathcal{D} modulo I , denoted by \mathcal{D}^{-I} , is the network \mathcal{D} reduces to, given the partial interpretation I . It contains the remaining part of the

network after the influence of I on the remaining nodes of \mathcal{D} has been taken into account.

We give the corresponding formal definitions :

Definition 8.1 *The network \mathcal{D} restricted to I is the network $\mathcal{D}_I = (N_I, J_I)$, where $N_I = \text{Ker}(I)$ and $J_I = \{j \in J \mid At(j) \subseteq \text{Ker}(I)\}$* ■

Definition 8.2 *The network \mathcal{D} modulo I is the network $\mathcal{D}^{-I} = (N^{-I}, J^{-I})$, where $N^{-I} = N - \text{Ker}(I)$ and J^{-I} is the set of justifications*

$$j' = (\alpha - I_t) \wedge (\beta - I_f) \rightarrow c$$

for which there exists a justification $j = \alpha \wedge \beta \rightarrow c \in J$ such that $I(c) = u$ and $I(\alpha \wedge \beta) >_t f$. ■

The skeptical model of a basic RMS \mathcal{D} can be shown to be equivalent to the least fixed point of the operator $TT_{\mathcal{D}}$ defined as :

$$TT_{\mathcal{D}}(I) = I \sqcup (M_1(\mathcal{D}^{-I}), N^{-I} - M_2(\mathcal{D}^{-I}))$$

where

- $M_1(\mathcal{D}^{-I})$ is the least 2-valued model of the monotonic network $\mathcal{D}_1^{-I} = (N^{-I}, J_1^{-I})$ with $J_1^{-I} = \{j \mid j = \alpha \rightarrow c \in J^{-I}\}$ and
- $M_2(\mathcal{D}^{-I})$ is the least model of the monotonic network $\mathcal{D}_2^{-I} = (N^{-I}, J_2^{-I})$, with $J_2^{-I} = \{\alpha \rightarrow c \mid \alpha \wedge \beta \rightarrow c \in J^{-I}\}$

Let $TT_{\mathcal{D}}^{\uparrow 0} = \emptyset^2$ and $TT_{\mathcal{D}}^{\uparrow k+1} = TT_{\mathcal{D}}(TT_{\mathcal{D}}^{\uparrow k})$. Then we have :

Theorem 8.3 *$\text{lfp}(TT_{\mathcal{D}})$ is the unique \sqsubseteq_k -minimal stable partial model of \mathcal{D} .*

The skeptical model can be obtained very efficiently. Note that every iteration of $TT_{\mathcal{D}}$ requires the computation of two minimal models of *monotonic* subnetworks of \mathcal{D} .

By a result of Dowling and Gallier ([5]), such a minimal model of a monotonic (or Horn) theory can be found in linear time ($O(|\mathcal{D}|)$). Since N is finite and there are at most N iterations of $TT_{\mathcal{D}}$, $\text{lfp}(TT_{\mathcal{D}})$ can be found in $O(|N| \times |\mathcal{D}|)$ time.

Part II

Invited Presentations

Nonmonotonic inferences based on expectations: A preliminary report

Peter Gärdenfors
 Cognitive Science
 Department of Philosophy
 University of Lund
 S-223 50 Lund, Sweden
 E-mail: Peter.Gardenfors@fil.lu.se

Abstract

The key idea is that when we want to find out whether α nonmonotonically entails β , the background information used to determine this does not only consist of full beliefs but also of (defeasible) expectations. If the set of expectations (including the beliefs) is regarded as an assumed state of belief, then the problem of nonmonotonic inference can be regarded as a problem of belief revision since the expectations can be defeated by the premiss α . On the basis of this correspondence, representation theorems from the logic of belief revision can be translated to produce new representation results for nonmonotonic logics. It can be shown that the logics obtained from the expectation semantics are closely related to those studied by Shoham and Kraus, Lehmann and Magidor.

1 MOTIVATION¹

In this article I want to argue that different forms of nonmonotonic inferences can be given a unified treatment in terms of how *expectations* are used in reasoning. The guiding idea is that when we try to find out whether β follows from α , the background information that we use for the inference does not only contain what we firmly believe, but also information about what we *expect* in the given situation. Such expectations can be expressed in different ways: by default assumptions, statements about what is normal or typical, etc. These expectations are not

full beliefs but *defeasible* in the sense that if the premise α is in conflict with some of the expectations, we do not use them when determining whether β follows from α .

I want to argue that expectations are the same kind of information as 'full' beliefs; the difference is that they are more defeasible than 'full' beliefs. This is one side of the unified treatment of nonmonotonic reasoning. For simplicity I shall work with a standard propositional language L including the classical connectives.

The expectations will be described as a set Δ of sentences from the language L . We also assume that the full beliefs are included in Δ , since what we believe we expect to be true.

The key idea can be put informally as follows:

α nonmonotonically entails β iff β follows logically from α together with as many as possible of the sentences in Δ that are compatible with α .

In order to make this more precise, we must, of course, specify what is meant by 'as many as possible'.² But before we turn to technicalities, let me illustrate the gist of the analysis by an example. ' α nonmonotonically entails β ' will be denoted $\alpha \vdash \beta$ as usual.

Let the language L contain the following predicates:

- Sx: x is a Swedish citizen
- Ix: x has Italian parents
- Px: x is a protestant

Assume that Δ contains the expectations $Sx \rightarrow Px$ and $Sx \& Ix \rightarrow \neg Px$.³ Since we have assumed that Δ is closed

²This idea is related to the idea of 'minimal change' within the theory of belief revision (see Gärdenfors (1988), pp. 66-68).

³Open formulas are assumed to be logically equivalent with their closures. I omit quantifiers, since they tend to muddle the notation.

¹This paper reports ongoing research that I am doing in collaboration with David Makinson. The full version will be a joint paper.

under logical consequences, it also contains $Sx \rightarrow \neg Ix$ and, of course, the logical truth $Sx \& Ix \rightarrow Sx$. If we now learn that a is a Swedish citizen, that is Sa , this piece of information is consistent with Δ and thus we can conclude that $Sa \vdash Pa$ according to the recipe above.

On the other hand, if we learn both that a is a Swedish citizen and has Italian parents, that is, $Sa \& Ia$, then this information is *inconsistent* with Δ and so we cannot use all expectations in Δ when determining which inferences can be drawn from $Sa \& Ia$. The most natural expedient is to give up the expectation $Sx \rightarrow Px$ and the consequence $Sx \rightarrow \neg Ix$. The contracted set of expectations, which contains $Sx \& Ix \rightarrow \neg Px$ and its logical consequences in a sense (to be made precise below), contains as many sentences as possible in Δ that are compatible with $Sa \& Ia$. So, by the general rule above, we have $Sa \& Ia \vdash \neg Pa$. This shows that \vdash is indeed a nonmonotonic inference operation.

The main technical problem to be solved is to present a more rigorous formulation of the general criterion above. This will be the objective of Sections 2 and 5.

2 EXPECTATION INFERENCE OPERATIONS BASED ON MAXIMAL SUBSETS

In this paper I shall work with a language L which is based on propositional logic. The details of L will be left open for the time being. It will be assumed that L is closed under *boolean operators*. I will use $\alpha, \beta, \gamma, \dots$ as variables over sentences in L . It is also convenient to introduce the symbols T and \perp for the two sentential constants "truth" and "falsity."

All the different expectations will be formulated in L . In contrast to many other theories of nonmonotonic reasoning there are thus no default rules or other additions to the basic language that will be used to express the defeasible forms of information.

In general we will assume that the underlying logic includes *classical propositional logic* and that it is compact. If Δ logically entails α , we will write this as $\Delta \vdash \alpha$. Where Δ is a set of sentences, we shall use the notation $Cn(\Delta)$ for the set of all logical consequences of Δ , i.e., $Cn(\Delta) = \{\alpha : \Delta \vdash \alpha\}$.

Several nonmonotonic inference operations will be studied. The fact that α nonmonotonically entails β will be denoted $\alpha \vdash \beta$. We will also introduce the notation $C(\alpha)$ for the set of all nonmonotonic conclusions that can be drawn from α , that is, $\beta \in C(\alpha)$ iff $\alpha \vdash \beta$.

The set of 'expectations' will be denoted Δ . In the main part of the paper it will be assumed that Δ is consistent and *closed under logical consequences*, i.e., $\Delta = Cn(\Delta)$. The problem in focus is how to define which elements of the set Δ to give up when adding a new piece of information α that is inconsistent with Δ . A general idea is to start from Δ and then give some recipe for choosing which propositions to delete from Δ to form a subset that

does not contain α as a logical consequence. According to the general criterion we should look at as *large* a subset of Δ as possible.

The following notion is useful: A set D is a *maximal subset of Δ that fails to imply α* if and only if (i) $D \subseteq \Delta$, (ii) $\alpha \notin D$, and (iii) for any sentence β that is in Δ but not in D , $\beta \rightarrow \alpha$ is in D . The last clause means that if D were to be expanded by β it would entail α . The set of all maximal subsets that fail to imply α will be denoted $\Delta \perp \alpha$.

We now turn to a first solution to the problem of determining when $\alpha \vdash \beta$ holds, that is, determining when α nonmonotonically implies β . The idea is to use the sets in $\Delta \perp \alpha$ for the construction since these sets are consistent with α . Technically, this can be done with the aid of a *selection function* S over Δ , such that $\emptyset \neq S(\Delta \perp \alpha) \subseteq \Delta \perp \alpha$, when $\Delta \perp \alpha$ is non-empty, and $S(\Delta \perp \alpha) = \Delta$ otherwise.

Definition 1: An expectation inference operation $C_{\Delta,S}$ is defined, for all $\alpha \in L$, by the equation $C_{\Delta,S}(\alpha) = \cap\{Cn(\{\alpha\} \cup D) : D \in S(\Delta \perp \alpha)\}$, where Δ is a non-empty default set and S is a selection function. $C_{\Delta,S}$ is *closed* when $\Delta = Cn(\Delta)$.

Intuitively, when we want to determine whether $\alpha \vdash_{\Delta,S} \beta$, that is, whether $\beta \in C_{\Delta,S}(\alpha)$, we consider the 'most relevant' maximal subset of Δ that fails to imply $\neg \alpha$ which are picked out by the selection function S . If $\alpha \rightarrow \beta$ belongs to *all* of these maximal subsets, then β belongs to $Cn(\{\alpha\} \cup D)$ for all D in $S(\Delta \perp \alpha)$ and so, by the definition, $\alpha \vdash_{\Delta,S} \beta$. It should be noted that expectation inference operations are here only defined for *finite* sets of premises, which can always be replaced by their conjunction α , which is the single formula in the definition above.

Definition 1 proffers a wide class of inference operations because no particular constraints are put on the selection function. I shall now give a characterization of the class of expectation inference operations by examining which general conditions on inference operations they satisfy. The conclusion of this investigation will be a representation theorem.

3 POSTULATES FOR NONMONOTONIC INFERENCE OPERATIONS

In this section I present some axioms for nonmonotonic inference operations. One restriction of my presentation is that I shall only be working with *finite* sets of premises in my formulation of the axioms.

$$\alpha \vdash \alpha \\ (\text{Reflexivity})$$

$$\text{If } \vdash \alpha \leftrightarrow \beta \text{ and } \alpha \vdash \gamma, \text{ then } \beta \vdash \gamma \\ (\text{Left Logical Equivalence})$$

$$\text{If } \vdash \beta \rightarrow \gamma \text{ and } \alpha \vdash \beta, \text{ then } \alpha \vdash \gamma \\ (\text{Right Weakening})$$

If $\alpha \vdash \beta$ and $\alpha \vdash \gamma$, then $\alpha \vdash \beta \& \gamma$
(And)

The following rule can be used to replace the conjunction of Right Weakening and And:

If $\alpha \vdash \beta_i$ for all $\beta_i \in B$ and $B \vdash \gamma$, then $\alpha \vdash \gamma$
(Closure)

Lemma 1: Closure is equivalent to the conjunction of Right Weakening and And.

If $\alpha \vdash \perp$, then $\alpha \vdash \perp$
(Consistency Preservation)

This condition is not considered by Kraus, Lehmann, and Magidor (1990), but it emerges as the translation of the postulate (K*5) for belief revision in Makinson and Gärdenfors (1990).

If $\alpha \vdash \beta$, then $\vdash \alpha \rightarrow \beta$
(Weak Conditionalization)

If $\not\vdash \neg\alpha$ and $\vdash \alpha \rightarrow \beta$, then $\alpha \vdash \beta$
(Weak Rational Monotony)

Makinson and Gärdenfors (1990) translated the postulates (K*1) - (K*8) taken from Gärdenfors (1988) for belief revision into postulates for nonmonotonic operations. The key idea is that a statement of the form $\beta \in K^* \alpha$, where $K^* \alpha$ is the revision of a belief state K by a sentence α , is seen as a nonmonotonic inference from α to β given the set K of sentences as *background* (default) information. So the statement $\beta \in K^* \alpha$ for belief revision is translated into the statement $\alpha \vdash \beta$ for nonmonotonic logic (or into $\alpha \vdash_K \beta$, if one wants to emphasize the role of the background beliefs). It turns out that the translations of the postulates (K*1) - (K*6) correspond precisely to Closure, Reflexivity, Weak Conditionalization, Weak Rational Monotony, Consistency Preservation, and Left Logical Equivalence, respectively (see Makinson and Gärdenfors (1990), Section 4). Thus, given lemma 1, this set of translations is equivalent to the set of postulates introduced so far. This set will be called the *basic* postulates.

However, it can be shown by simple examples that from the basic postulates one cannot derive the following principle which has occupied a central place in the discussion of nonmonotonic logics (Makinson 1989, 1991):

If $\alpha \vdash \beta$ and $\beta \vdash \alpha$, then $\alpha \vdash \gamma$ iff $\beta \vdash \gamma$
(Cumulativity)

Cumulativity is equivalent to the conjunction of the following two postulates which are perhaps more perspicuous:

If $\alpha \vdash \beta$ and $\alpha \& \beta \vdash \gamma$, then $\alpha \vdash \gamma$
(Cut)

If $\alpha \vdash \beta$ and $\alpha \vdash \gamma$, then $\alpha \& \beta \vdash \gamma$
(Cautious Monotony)

Furthermore, given Reflexivity, Cumulativity is equivalent to the following useful postulate:¹

If $\alpha \vdash \beta$ and $\beta \vdash \alpha$, then $\alpha \vdash \gamma$ iff $\beta \vdash \gamma$
(Reciprocity)

The final postulates are more controversial and satisfied by a smaller number of nonmonotonic systems.²

If $\alpha \vdash \gamma$ and $\beta \vdash \gamma$, then $\alpha \vee \beta \vdash \gamma$
(Or)

This postulate is equivalent to the following, given the basic postulates:³

If $\alpha \& \beta \vdash \gamma$, then $\alpha \vdash \beta \rightarrow \gamma$
(Conditionalization)

Conditionalization is the translation in Makinson and Gärdenfors (1990) of the postulate (K*7) for belief revision. Weak Conditionalization is the special case when $\alpha = T$.

The following postulate is studied by Kraus, Lehmann, and Magidor (1990) and Lehmann and Magidor (1989), and it is equivalent to the translation in Makinson and Gärdenfors (1990) of the postulate (K*8) for belief revision:

If $\alpha \not\vdash \neg\beta$ and $\alpha \vdash \gamma$ then $\alpha \& \beta \vdash \gamma$
(Rational Monotony)

It can easily be shown that Rational Monotony and Conditionalization together with the basic postulates entail Cumulativity.

4 REPRESENTATION THEOREMS FOR SELECTION FUNCTIONS

In this section it will be shown that the set of basic postulates for nonmonotonic inference operations exactly characterizes the class of expectation inference operations.

Theorem 1: A nonmonotonic inference operation \vdash satisfies Closure, Reflexivity, Weak Conditionalization, Weak Rational Monotony, Consistency Preservation, and Left Logical Equivalence (i.e., the set of basic postulates) if and only if there exists a closed expectation inference operation $\vdash_{\Delta, S}$ such that $\alpha \vdash \beta$ iff $\alpha \vdash_{\Delta, S} \beta$, for all α and β .⁴

In this theorem no restrictions are put on the selection function. It shows that for nonmonotonic inference operations based on such selection functions we can, in general, only expect the basic postulates to be satisfied, but, for example, not Cumulativity. A very natural condition on a selection function is the following:

¹For discussions of these postulates and proofs, see Makinson (1991).

²For a survey of nonmonotonic systems and which postulates they satisfy, see Makinson (1991).

³See Makinson (1991) for a proof.

⁴Proofs of theorems and lemmas will be given in the full version of the paper.

- (SC) If $S(\Delta \perp \neg \alpha) \subseteq \Delta \perp \neg \beta \subseteq \Delta \perp \neg \alpha$,
then $S(\Delta \perp \neg \beta) = S(\Delta \perp \neg \alpha)$

The interpretation is that if the set of maximal subsets in $\Delta \perp \neg \beta$ are included in the set $\Delta \perp \neg \alpha$ and the 'preferred' maximal subsets in $\Delta \perp \neg \alpha$ are all members of $\Delta \perp \neg \beta$, then these are also the best in $\Delta \perp \neg \beta$.

Theorem 2: A nonmonotonic inference operation \vdash satisfies the set of basic postulates and Cumulativity if and only if there exists a closed expectation inference operation $\vdash_{\Delta, S}$ where S satisfies (SC) such that $\alpha \vdash \beta$ iff $\alpha \vdash_{\Delta, S} \beta$, for all α and β .

5 EXPECTATION ORDERINGS

The use of selection functions as a mechanism for generating nonmonotonic inferences is not very satisfactory from a computational perspective. The reason is, first of all, that it is extremely costly to compute the maximal subsets in $\Delta \perp \neg \alpha$, and secondly, a functional description of a selection function is, in general, not available.

It would be much more natural to work with an ordering of the *sentences* in Δ rather than with an ordering of the maximal subsets of Δ . And from the epistemological perspective it seems intuitively plausible that our expectations about the world do not all have the same strength. In brief, the expectations in Δ are all defeasible (unless logically valid), but they exhibit varying *degrees of defeasibility*.

When determining whether α nonmonotonically entails β , the different degrees of expectation among the sentences in Δ can then be used to determine which sentences to give up from Δ when α is inconsistent with Δ .

In order to make this idea more precise, I shall now assume that there is an ordering \leq of the sentences in L .¹ ' $\alpha \leq \beta$ ' should be interpreted as ' α is less expected than β '. ' $\alpha < \beta$ ' will be written as an abbreviation for 'not $\beta \leq \alpha$ ' and ' $\alpha = \beta$ ' is an abbreviation for ' $\alpha \leq \beta$ and $\beta \leq \alpha$ '. The relation \leq will be assumed to satisfy certain constraints. The constraints to be presented here are exact parallels of the postulates (EE1) - (EES) for *epistemic entrenchment* presented in Gärdenfors (1988).

- (E1) If $\alpha \leq \beta$ and $\beta \leq \gamma$, then $\alpha \leq \gamma$
(Transitivity)
- (E2) If $\alpha \vdash \beta$, then $\alpha \leq \beta$
(Dominance)
- (E3) For any α and β , $\alpha \leq \alpha \& \beta$ or $\beta \leq \alpha \& \beta$
(Conjunctiveness)
- (E4) If Δ is consistent, $\alpha \notin Cn(\Delta)$ iff $\alpha \leq \beta$ for all β
(Minimality)

- (E5) $\beta \leq \alpha$ for all β , only if $\vdash \alpha$
(Maximality)

Lemma 2: (E1) - (E3) implies that \leq is connected, that is, $\alpha \leq \beta$ or $\beta \leq \alpha$ for all α and β .

Let us now return to how the ordering \leq can be used to determine when α nonmonotonically implies β . According to the general recipe of this paper, $\alpha \vdash \beta$ means β follows from α together with as many of the expectations in Δ as possible that are compatible with α . A first attempt is to use a direct translation of the corresponding condition for contraction functions in Gärdenfors and Makinson (1988). This leads, more or less directly, to the following criterion:

- (Ch \sim) $\alpha \vdash \beta$ iff $\neg \alpha \rightarrow \beta < \alpha \rightarrow \beta$ or $\vdash \neg \alpha$.

However, David Makinson has shown that the following much more intuitive definition will give equivalent results:

Definition 2: \vdash is an *expectation-based* nonmonotonic inference relation iff there is a consistent set Δ and an ordering \leq satisfying (E1) - (E5) such that the following condition holds:

- (Ch \sim) $\alpha \vdash \beta$ iff $\beta \in Cn(\{\alpha\} \cup \{\beta: \neg \alpha < \beta\})$

The idea behind the condition (Ch \sim) is that when determining whether $\alpha \vdash \beta$, one looks at what follows from α together with the class of sentences in Δ that are more expected than $\neg \alpha$.

It can now be shown that an expectation-based nonmonotonic inference relation defined by (Ch \sim) can be characterized in a very strong way:

Theorem 3: \vdash is an expectation-based nonmonotonic inference relation if and only if \vdash satisfies the basic postulates together with Distribution and Rational Monotony.

The interpretation of the strong part of this theorem is that if you believe that basic postulates together with Distribution and Rational Monotony are reasonable for a nonmonotonic inference relation \vdash , then there will exist an underlying expectation relation \leq of the sentences of L , satisfying (E1) - (E5), that will generate exactly the inference relation \vdash .

6 RELATIONS TO PREFERENTIAL MODELS

I want to show in this section that there is a very close correspondence between expectation orderings and *preferential models* in the style of Shoham (1988).

Let us introduce a variant of Shoham's preferential models. For the definition, let M be the set of all maximal consistent subsets of L , i.e., the set of *models*. $|M|$ denotes the set of models where α is true, i.e., $x \in |M|$ iff $\alpha \in x$. Similarly, if Γ is a set of sentences, $|\Gamma|$ denotes the set of models where all sentences in Γ are true.

¹Another way of generating expectations underlying nonmonotonic inferences is presented in Balkenius and Gärdenfors (1991).

Definition 3: A preferential model W is a pair $\langle \leq, \Delta \rangle$, where Δ is a consistent set of sentences and \leq is a relation on M satisfying the following postulates:

- (P1) \leq is transitive
- (P2) \leq is connected
- (P3) If $|\alpha| \neq \emptyset$, then $\{x \in |\alpha| : x \leq y \text{ for all } y \in |\alpha|\} \neq \emptyset$
- (P4) $x \in M$ is \leq -minimal iff $x \in |\Delta|$

The main difference between Shoham's models and the ones presented here is that he only assumes that \leq is a partial order, while I assume modularity as well, which means that \leq is connected. Note that the smoothness condition in Definition 3.8 of Kraus, Lehmann, and Magidor (1990) follows from (P3'). However, it should be noted that I nowhere need the distinction between *states* and *worlds* that complicates the structure of their models.

Let us denote the set $\{x \in |\alpha| : x \leq y \text{ for all } y \in |\alpha|\}$ by $p(\alpha)$. This is the set of *most preferred* α -models.

Definition 4: \vdash_p is a *preferential* nonmonotonic inference relation iff there is a preferential model W with a consistent set Δ and an ordering \leq satisfying (P1) - (P4) such that the following condition holds:

$$(C\vdash_p) \alpha \vdash_p \beta \text{ iff } x \in |\beta| \text{ for all } x \in p(\alpha).$$

Before stating the connections between Δ -preferential models and expectation-based nonmonotonic inference relations, let me introduce three very useful notions: Firstly, a set Γ of sentences is a *cut* of an expectation ordering \leq iff $\beta \in \Gamma$ whenever $\alpha \in \Gamma$ and $\alpha \leq \beta$, for all α and β . It follows from the postulates (E4) and (E5) in particular that the largest cut is the set L , the second largest is $Cn(\Delta)$ and the smallest is the set of all logical truths.

By (E2) and (E1) it follows that cuts are closed under logical consequences. Furthermore, the connectivity of \leq entails that cuts are nested, i.e., for any cuts Γ and Σ , either $\Gamma \subseteq \Sigma$ or $\Sigma \subseteq \Gamma$. Given this, it is easy to show that the set of cuts are closed under intersections and unions. By using the crucial property (E3), we can also prove:

Lemma 3: for any cut Γ and any sentence α , $\alpha \in \Gamma$ iff $|\Gamma| \subseteq |\alpha|$.

Secondly, let Γ_α be the union of all cuts Γ such that $\neg\alpha \notin \Gamma$. Since cuts are closed under unions, Γ_α is itself a cut such that $\neg\alpha \notin \Gamma_\alpha$ and hence by the lemma above $|\Gamma_\alpha| \not\vdash \neg\alpha$. Furthermore, if not $\vdash \neg\alpha$, then $\Gamma_\alpha \neq \emptyset$.

Thirdly, if $|\alpha| \neq \emptyset$, put $c(\alpha) = \{z \in M : z \leq x \text{ for all } x \in p(\alpha)\}$. An equivalent definition is $c(\alpha) = \{z \in M : z \leq x \text{ for all } x \in |\alpha|\}$. When $|\alpha| = \emptyset$, we define $c(\alpha) = \emptyset$.

I now turn to the connections between the two kinds of ordering. We first show how to turn an expectation

ordering into a preferential model. To accomplish this we need a definition of \leq in terms of \leq :

$$(\text{Def } \leq) x \leq y \text{ iff for all cuts } \Gamma, \text{ if } y \in |\Gamma|, \text{ then } x \in |\Gamma|$$

Lemma 4: If the ordering \leq of M is defined by (Def \leq), then $c(\alpha) = |\Gamma_\alpha|$.

Corollary: For all α and β , $c(\alpha) \subseteq c(\beta)$ or $c(\beta) \subseteq c(\alpha)$.

Theorem 4: Let \leq be an expectation ordering satisfying (E1) - (E5). Then the ordering \leq of M defined by (Def \leq), is a preference ordering satisfying (P1) - (P4). Furthermore, the preferential nonmonotonic inference relation generated from \leq is identical with the expectation-based nonmonotonic inference relation generated from \leq .

We next turn to the converse problem, that of using a presentational model to construct an expectation ordering. The appropriate definition of \leq in terms of \leq is the following:

$$(\text{Def } \leq) \alpha \leq \beta \text{ iff } c(\neg\alpha) \subseteq c(\neg\beta)$$

Theorem 5: Let \leq be a preference ordering satisfying (P1) - (P4). Then the ordering \leq of L defined by (Def \leq) is an expectation ordering satisfying (E1) - (E5). Furthermore, the expectation-based nonmonotonic inference relation generated from \leq is identical with the preferential nonmonotonic inference relation generated from \leq .

By combining these results with Theorem 3, we obtain the following:

Corollary: \vdash is a *preferential* nonmonotonic inference relation iff \vdash satisfies the basic postulates together with Distribution and Rational Monotony.

Acknowledgements

Research for this article has been supported by the Swedish Council for Research in the Humanities and Social Sciences. David Makinson's comments have, as usual, been tremendously inspiring. I also wish to thank Sten Lindström for helpful comments.

References

- Alchourrón, C. E., P. Gärdenfors, and D. Makinson (1985): "On the logic of theory change: Partial meet contraction and revision functions," *The Journal of Symbolic Logic* 50, 510-530.
- Balkenius, C. and Gärdenfors P. (1991): "Nonmonotonic inferences in neural networks," in these *Proceedings*.
- Gärdenfors, P. (1988): *Knowledge in Flux: Modeling the Dynamics of Epistemic States* (Cambridge, MA: The MIT Press, Bradford Books).
- Gärdenfors, P. and D. Makinson. (1988): "Revisions of knowledge systems using epistemic entrenchment," in *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, M. Vardi ed. (Los Altos, CA: Morgan Kaufmann).

Kraus, S., D. Lehmann, and M. Magidor, (1990): "Nonmonotonic reasoning, preferential models and cumulative logics," *Artificial Intelligence* 44, 167-207.

Makinson, D. (1989): "General theory of cumulative inference," in M. Reinfrank, J. de Kleer, M.L. Ginsberg, and E. Sandewall, eds., *Non-Monotonic Reasoning* (Berlin: Springer Verlag, Lecture Notes on Artificial Intelligence no 346).

Makinson, D. (1991): "General patterns in nonmonotonic reasoning," to appear as chapter 2 of *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume II: Non-Monotonic and Uncertain Reasoning*. (Oxford: Oxford University Press).

Makinson, D. and P. Gärdenfors (1990): "Relations between the logic of theory change and nonmonotonic logic," to appear in the proceedings of the Konstanz workshop on belief revision, ed. by A. Fuhrmann and M. Morreau (Berlin: Springer Verlag).

Lindström, S. (1990): "Basic nonmonotonic logic," manuscript, Department of Philosophy, Uppsala University.

Shoham, Y. (1988): *Reasoning about Change*. (Cambridge: Cambridge University Press).

Implemented Knowledge Representation and Reasoning Systems: An Endangered Species?

(*Extended Abstract of Invited Talk*)

Charles Rich
 Artificial Intelligence Laboratory
 Massachusetts Institute of Technology
 Cambridge, MA 02139

On March 26–28, 1991, a group of system implementors, users, and theoreticians (not disjoint categories) will meet as part of the AAAI Spring Symposium Series to discuss the role of *implemented* systems in knowledge representation and reasoning research. A total of twenty-two implemented systems will be represented at the workshop (see Table 1).

The motivation for this meeting is the feeling, at least on the part of the organizers, that there is a present and growing divergence in the field between theoretical work and system building. In particular, the trend in current published work seems to greatly favor the purely theoretical analysis of unimplemented formal systems. This is a serious problem for the field in the long run because, as in any science, theory is most productive when it advances hand in hand with implementation, experimentation, and practice.

The root of the problem, we feel, is the lack of a paradigm (in the sense of (Kuhn, 1972)) that connects theory and experiment. (Perhaps Cohen's (1991) terminology, "models and systems," is more appropriate to our field). In the absence of a codependence between these two kinds of work, the purely theoretical investigations will always predominate because the implementation of systems is so much more costly in terms of both human and computer resources.

We hope this workshop will help us begin to forge such a paradigm. One of the key purposes of a scientific paradigm is to define what constitutes a "result." A shared paradigm is necessary to allow researchers to communicate effectively with each other, evaluate progress, and build on each other's work. The direction our thinking in this regard is reflected in Table 2, which is a list of questions that each system implementor will be asked to answer.

Finally, it is worth observing that the methodological tension we describe here is also being felt more broadly in the AI research community (see Cohen, 1991).

Acknowledgements

I would like to thank the rest of the workshop organizing committee, Ron Brachman, Marc Vilain, Stuart Shapiro, and Ramesh Patil.

References

- P. R. Cohen. (1991) A Survey of the Eighth National Conference on Artificial Intelligence: Pulling Together or Pulling Apart? *AI Magazine* 11(4).
- T. S. Kuhn. (1970) *The Structure of Scientific Revolutions*. Chicago, IL: University of Chicago Press.

Table 1: Systems Represented at the Workshop

System	Affiliation
RHET	U. of Rochester
KRIS	DFKI, Germany
KING KONG	MITRE
LILOG	IBM, Germany
IMMORTAL	U. of Illinois, Urbana-Champaign
ALGERNON	U. of Texas, Austin
KRS	U. of Calgary
MVL	Stanford U.
ITL	LADSEB-CNR, Italy
SB-ONE	U. Saarbrucken
TELOS	U. Toronto
CYCL	MCC
LOOM	USC/ISI
K-REP	IBM, New York
ONTIC	MIT
JOSIE	Price Waterhouse
CLASSIC	AT&T Bell Labs
BACK	TU, Berlin
CAKE	MIT
SNEPS	SUNY, Buffalo
JOSHUA	Symbolics
EPIKIT	Epistemics

Table 2: Questions to Ask about an Implemented System

Why, Why, Why, ...

- What was the motivation for building the system? If there is more than one motivation, can you prioritize them?
- Did you consider building on or adapting another system as an alternative to building one from scratch? If so, which other systems did you consider, why did you consider each one, and what was your conclusion in each case?

What's It Good For?

- Is the system intended to be used for some application? If so, can you characterize the kinds of problems it is intended to be used for independently of the system?
- Is there a target user community? What has been your interaction with that community?

How Does It Measure Up?

- Generally speaking, to what degree do you feel the system has achieved the goals that motivated it?
- What properties (if any) of the system do you measure or formally evaluate in any way? For example, do you have some measure of system performance?
- Do you have some standard examples that you use in working on the system? How did you come up with them? How do you use them? Would they be appropriate for other systems? If not, why not?
- What other systems does it make sense to compare the system with and, in each case, what kind of comparison is appropriate?

Models and Predictions

- Do you use any formal models of either the system behavior or the problems it solves? If so, please describe them briefly.
- Can you predict, using one of the models above, how certain design changes will affect the behavior of the system? If so, please give a few examples.
- Can you predict, using one of the models above, how system performance will be affected by certain variations in the problem space? If so, please give a few examples.
- Can you identify one or more algorithms that are central to the operation of the system? If so, please describe them briefly. Are any of these algorithms novel?

Report from the Trenches

- What parts or aspects of the system have consumed the most implementation resources?
- Are there parts of the system implementation that failed to behave properly and were subsequently discarded (or kept)? If so, describe them briefly along with why they failed.
- What parts of the system needed to be optimized? Why? How did you find out?
- Did you explicitly explore engineering tradeoffs in any parts of the system design, such as time versus space? If so, describe them briefly.
- What implementation foundation (e.g., programming language) did you choose? Why? Did you consider alternatives and choose based on some technical differentiation (if so, what was it)?

Lessons Learned

- What aspects of the system are you most proud of and would you insist on doing just the same way the next time?
- Do you have any advice for those building similar systems regarding what *not* to do? What design decisions would you make differently? (Try to rank these in order of importance and impact.)

Publish or Perish

- Have you published any papers about the system? If so, in what conferences and/or journals? If not, have you submitted papers and had them rejected? If so, to what conferences and/or journals? Why were they rejected?
- Can you cite any published papers in AI or elsewhere in computer science that you feel are role models for how to write a good paper about an implemented knowledge representation and reasoning system? Please suggest both conference and journal papers (if you know of any).
- What types or groups of people do you think would or should be interested in the system, and why? In particular, what benefit do you think the system could be to people other than those who might actually use it?

Part III

Position Papers from the Panel on Knowledge Sharing and Reuse

Panel: Achieving Large Scale Knowledge Sharing

William Mark
Information and Computing Sciences
Lockheed Palo Alto Research Labs
3251 Hanover St. O/96-01 B/254E
Palo Alto, CA 94304-1191
(415)354-5236 FAX (415)354-5235
mark@sumex.stanford.edu

Abstract

There is a growing sense (at least in some quarters) that the longstanding goal of large knowledge bases is now within reach. However, even among the optimists, there are differing views about the key technical and non-technical issues that must be addressed if we are to achieve large knowledge bases. This panel will explore some of the opinions, concentrating on various views of the knowledge representation problems and the various proposed research approaches to their solution.

Large scale knowledge bases have been a goal of AI for many years (probably since the inception of the field). For one thing, if an AI system is to perform a significant task such as understanding a substantial subset of natural language or acting autonomously in a real world context, it requires a large amount of knowledge. Also, there is the dream of "national knowledge bases" to be built and shared by a large community of practitioners (say, engineers or doctors). Another version is the enterprise-wide, or even project-wide, knowledge base to support decision making by various stakeholders with different interests and different expertise.

These visions place the knowledge in varying roles: as a repository of problem solving know-how in which the right information is located and used; as an integral record that supports reasoning about the current

set of decisions that have been made about something; as a medium of communication, a common set of terms and policies, to be used by people cooperating to solve a problem; and so on.

No matter what the envisioned role for large knowledge bases, there are some important and difficult issues to be addressed: technical (how to build, access, and maintain large scale knowledge bases), sociological (who builds the knowledge base, and who has access to it? how will the effort to build large scale knowledge bases affect knowledge representation research as a whole?), legal (how will intellectual property rights be handled?), and so on.

At least some of the technical issues involved in large knowledge bases are clear:

- Constructing large knowledge bases will require a sustained effort, probably by many people, over a long period of time. There must be some automatic means (or at least some very effective guidance) for *accumulating* the knowledge, i.e., taking knowledge provided in incomplete "pieces" and fitting it into the framework of existing knowledge.
- Using large knowledge bases, by people or programs, requires some concept of *indexing*, i.e., a means to find relevant knowledge without having to examine it detail.

Other issues are of course equally "clear", but the central point is that *sharing* knowledge is essential to the construction and use of large knowledge bases.

Interest in shared knowledge bases has recently intensified due to advances in knowledge representation technology, the high visibility of the Cyc experiment at MCC, and widespread recognition that the current piecemeal approach to building knowledge bases is a barrier to further advances in some important areas of AI.

Some of the knowledge representation community see the technical issues of knowledge sharing as being at the heart of knowledge representation research. A growing number of researchers now believe that they see a path toward large scale knowledge sharing. To some extent, though, different researchers see different paths.

Some believe that there is currently (or soon will be) significant bodies of knowledge worth sharing, but that this knowledge is "trapped" in idiosyncratic knowledge representation systems used only by small groups (usually their developers). Therefore, the primary emphasis should be on opening the communication channels – developing knowledge representation formalisms capable of communicating knowledge among different knowledge representation systems. Others believe that sharable bodies of knowledge have yet to be built, and that in order to share knowledge, it must be built to be sharable in the first place. The primary emphasis should be on designing and representing sharable knowledge structures: formalisms for making explicit the assumptions that underlie the content and organization of the knowledge base, and mechanisms to continually determine and enforce the ramifications of changes to the knowledge base with respect to these assumptions. Still others see an intimate coupling between the knowledge representation and the use for which it is intended. The primary emphasis should be on analyzing the knowledge in terms of task dependencies, and "packaging" the knowledge along with its task dependent reasoning mechanisms in order to make it usefully sharable.

This panel will explore these (and other) technical issues relevant to achieving large scale knowledge sharing, focusing on the following questions:

To what extent is achieving large scale knowledge sharing a knowledge representation problem? Do we have the required knowledge representation technology? If not, what are the major areas requiring advance?

Knowledge Sharing: a Feasible Dream

Giuseppe Attardi
Dipartimento di Informatica
Università di Pisa
Corso Italia 40, I-56125 Pisa, Italy

1 HYPERKNOWLEDGE

We discuss an infrastructure for integrating AI software at the knowledge level, independent of particular implementations, which we call "HyperKnowledge": a means for describing, connecting and reusing knowledge-base components. Currently knowledge is expressed and communicated in written notation or through human communication, but its use by others requires an activity of human assimilation, adaptation and transfer into a new setting.

Communicating knowledge by means of HyperKnowledge, would allow knowledge to be directly applicable by others, to be extended and modified to adapt it to specific needs, and to be further communicated and linked with other knowledge.

A significant step forward in knowledge base technology is required that hinges upon the establishment of standards.

HyperKnowledge technology would support the creation and maintenance of shared knowledge bases synergistically integrated with software components within conventional systems:

1. it will provide rich, easily inspected sources of information that serve the functions of books and libraries but are more flexible, more easily updated, and easier to query;
2. it will enable the construction and marketing of pre-packaged knowledge services, allowing users to invoke, rent or buy services;
3. it will make it possible for end-users to create large systems attuned to their needs, constructing them and assembling them from reusable components.

HyperKnowledge systems would promote sharing and reuse of accumulated knowledge through mechanisms like:

1. libraries of multiple layers of reusable knowledge bases which can be either incorporated into software or consulted at execution time;
2. availability of standardized knowledge representation systems and means for translation between them;

3. tools and methodologies allowing developers to find and utilize both contents of those libraries useful to their needs, and pre-existing services built on those libraries complemented, by tools allowing developers to offer back their work for inclusion into the libraries.

2 TECHNICAL GOALS

The fundamental goal of widening the applicability of knowledge base techniques can be achieved through mechanisms which allow:

1. to construct sharable knowledge bases and reusable knowledge components;
2. to embed knowledge bases into larger software systems.

To achieve these goals, a technology must be developed which:

- offers libraries of reusable ontologies, consisting of: (1) formal definitions of the terms which can be used to model a domain or class of domains; and (2) assertions governing the requirements and constraints for creating valid models within a domain by combining and relating terms and term instances.
- offers powerful, expressive, and efficient interpreters and compilers for knowledge representation systems, in which those ontologies are embedded.
- provides system builders with tools for translating between alternative representation systems, enabling them to create efficient, optimized systems by making choices about alternative implementations.
- embeds those interpreters and compilers in architectures that support complete interoperability not just between multiple knowledge-based systems, but also with conventional software systems.

3 KNOWLEDGE MODELS

In the area of databases, several competing database models exist: relational, entity-relationship, object-oriented. Each model describes in abstract terms the structure and the operations supported by the database system. In the re-

lational database model, for instance, mathematical relations constitute the data model, projections and joins on relations are among the operations which constitute the functional interface to the database system, and finally the relational calculus is the query language used to express what information to derive or retrieve. It was only after the development of formal database model that relational database technology became of widespread use. Different implementations of the same model became available from different sources and could be compared and compete with each other. An *Abstract Knowledge Model* is essential to achieve similar effects of improving the ability of users to understand and contrast both differing models and differing implementations within the same model.

A knowledge model will consist of:

1. Knowledge Representation Model
2. Functional Library
3. Knowledge Query and Manipulation Language

For instance, in the case of term subsumption languages, the representational model consists of the structure of the terms, and their interconnection in the subsumption hierarchy. The functional specification consists in a series of operations, for instance to navigate through the subsumption hierarchy, to create new terms, to classify terms. The knowledge query language consists in a notation for expressing problems to the knowledge base.

An abstract knowledge representation model is important to improve the understanding of the model, its capabilities and limitations. Such model can focus research so as to develop helpful theoretical results providing insight, for instance on problems of complexity, tractability, and expressive power. Alternative implementations can be compared with respect to how well they support the functionalities of a model.

The primitives of the functional library provide support for knowledge representation services such as: storage/retrieval, unification/matching, propagation/mapping, navigation, and concomitant views.

The functional library would not be tied to any specific programming language, but language bindings should be provided for major programming languages. Specifying the primitives not as procedures but as a network protocol would be a solution that would fit quite well with a client/server view of a knowledge base system.

The Knowledge Query and Manipulation Language is a way to express interrogations to the Knowledge Base in a declarative way, leaving to the implementation to decide which operations are necessary.

4 SHARED ONTOLOGIES

Ontologies are ways of modelling the world, embodied by representational vocabulary and semantics for what makes sense to say in the vocabulary. Fostering the sharing of

ontologies is foremost to the integration of knowledge base systems. To achieve this, three things are needed:

- (1) creation of a public-domain set of common ontologies, knowledge bases built from those ontologies, and example applications built from those knowledge bases;
- (2) creation of a set of tools for inspecting, using, and extending ontologies and knowledge bases;
- (3) establishment of a public clearing-house for accumulation and dissemination of ontologies, knowledge bases, and tools, as well as for their ongoing support and maintenance.

5 TRANSFERABILITY

This concerns the possibility of transferring knowledge developed in a KB system to another. Each system works in its own way, but an external representation format is specified that each system can input and output. This interchange format will take the form of a full declarative language, like the proposed "Interlingua". Interlingua will have to be highly expressive in order to be capable of capturing the content of arbitrary representation languages, and will need a well-defined formal semantics in order to support translation.

6 INTEROPERABILITY

This concerns the possibility of directly invoking some service from a KB system and vice-versa. It assumes the possibility of (remote) procedure calls with transfer of data in their directly manipulatable internal representation. This is essential to allow developers of KB systems to produce packages or libraries which can be directly used or incorporated into the application, in as much as a statistical package or a graphical library can be linked and used by someone else.

So far, KB research has generally been based on programming languages whose traditional implementations limited the interaction with other languages, ruling out the possibility of building knowledge components which could be made available for embedding into other systems. For example, symbolic algebra systems, incorporating large amounts of knowledge about mathematics, could only be used as standalone systems, not as pieces to be assembled with other pieces. Similarly, one could imagine developing and making available packages for theorem proving, equation solving, program transformation, natural language processing, temporal reasoning and planning. Technical solutions to the interoperability problem are now available: networking protocols for remote services or Common Runtime Support for programming languages.

Knowledge Interchange Format

Michael R. Genesereth

Logic Group

Stanford University

Stanford, CA 94305

genesereth@cs.stanford.edu

The encoding of knowledge as data is an important first step toward widespread knowledge sharing. However, there are many ways in which this encoding can be done; and, if our programs are to share knowledge effectively, they must agree on this encoding.

One way to ensure this agreement is for us all, in writing our programs, to use a single knowledge representation system. In this way, we eliminate implementational differences and thereby facilitate the process of knowledge sharing. But which system should we use? If we choose one with an expressively limited language (such as primitive frames, slots, and values), we get computational efficiency, but we may be forced to go outside of the language to express certain types of knowledge (e.g. to encode the axiom of transitivity). If we choose a knowledge representation system with a general language (such as full first-order predicate calculus), we get expressiveness but only at the cost of computational overhead.

An alternative to this approach is to use different representations for different purposes but to agree on one (and exactly one) language for knowledge interchange. For internal storage and processing, we use structures, arrays, property lists, whatever we find convenient. For knowledge interchange, we use a general *interlingua* that permits the encoding of knowledge of different types. By providing encoders and decoders for this language, we make the knowledge in our programs available to other programs without requiring that those other programs conform to our idiosyncratic knowledge representation choices.

In March of 1990, DARPA sponsored a workshop in Santa Barbara aimed at investigating the possibilities of large-scale knowledge sharing. Despite the diversity of the attendees, the group came up with the beginnings of an agreeable knowledge transmission language.

Knowledge Interchange Format (KIF) is a formal language for the interchange of knowledge among disparate computer programs (written by different programmers, at different times, in different languages, and so forth). The language is a prefix form of predicate calculus augmented with quote and level-switching functions. There is a preliminary lexicon of common concepts. And there are procedures for expanding this lexicon.

KIF is *not* intended as a primary language for interaction with human users (though it can be used for this purpose). Different programs can interact with their users in whatever forms are most appropriate to their applications (for example frames, graphs, charts, tables, diagrams, natural language, etc.).

KIF is also *not* a standard for the internal representation of knowledge *within* computer programs or within closely related sets of programs (though it can be used for this purpose as well). Typically, when a program reads a database in KIF, it converts the data into its own internal form (specialized pointer structures, arrays, etc.). All computation is done using these internal forms. When the program needs to communicate with another program, it maps its internal data structures into KIF.

The purpose of KIF is roughly analogous to that of Postscript. Postscript is commonly used by text and graphics formatting programs in communicating information about documents to printers. Although it is not as efficient as a specialized representation for documents and not as perspicuous as a specialized what-you-see-is-what-you-get display, Postscript is a programmer-readable representation that facilitates the independent development of formatting programs and printers. While KIF is not as efficient as a specialized representation for knowledge nor as perspicuous as a specialized display (when printed in its

list form), it too is a programmer-readable standard and thereby facilitates the independent development of knowledge-manipulation programs.

The design of KIF satisfies the following rigorously evaluable criteria.

1. The language has descriptive semantics. It is possible to understand the meaning of expressions in the language without appeal to a procedure for manipulating those expressions. In this way, KIF differs from other languages that are based on procedural interpreters, such as Emycin and Prolog. (Note that this does not preclude the description of procedures within the language. In traditional programming languages, the description of a procedure is also a prescription for its use. In KIF, it is just a description. A program reading a such a description can choose to execute the procedure, to verify its functionality, to compute its running time, etc.).
2. The language is logically comprehensive – it provides for the expression of arbitrary sentences in nth-order predicate calculus. In this way, it differs from relational database languages (many of which are confined to ground atomic sentences) and Prolog-like languages (that are confined to Horn clauses).
3. The language is self-descriptive. It is possible describe the syntax and semantics of the language within the language itself. This allows the user to make all knowledge representation decisions explicit and permits the user to introduce new knowledge representation constructs without changing the language standard.

In addition to these hard criteria, KIF is designed to maximize in a joint fashion the following somewhat softer measures as well (to the extent possible while satisfying the preceding criteria).

1. Readability. Although the language is not intended as a language for interaction with humans, it *can* be “read” by human programmers.
2. Translatability. The language is an efficient formalism for communication, i.e. it does not consume an inordinate amount of space and it does not take an inordinate amount of time to encode or decode knowledge bases.
3. Tractability. Although the language is not intended for use within programs, it *can* be used by programs, if so desired, with at most logarithmic slowdown over specialized representations.

Unfortunately, the existence of a standard knowledge transmission language solves only a part of the problem of knowledge interchange. The knowledge encoded in one program, even when translated to KIF, is likely to be in a form best suited to the task of that program and the knowledge representation system it uses. Converting this knowledge for use on a different task or in a different knowledge representation system is likely to require great (and possibly prohibitive) cost.

As an example of the difficulties of putting knowledge to work on a new task, consider the job of building a diagnostician for a specific computer, given a knowledge base developed for simulation. While we can probably do this job by applying a model-driven diagnostician to the knowledge base, the resulting program is likely to be inefficient. In order to get acceptable efficiency, we might first need to reformulate the knowledge – e.g. into a set of diagnostic condition-action rules. Such rules can be expressed in KIF, but the use of KIF does not trivialize their derivation.

As an example of the difficulties of using a new knowledge representation system, consider the job of moving a knowledge base from a theorem-proving system to a frame-based system like Classic. The theorem prover’s knowledge base might implicitly define a class hierarchy, and this is just what we need to exploit the classification mechanisms of Classic. However, the axioms might be spread around in a way that makes the derivation of the class hierarchy very difficult.

(It is worth noting that similar problems would arise even if we were to use a single knowledge representation system. We would still have difficulties transforming knowledge for use on new tasks. And we would still have difficulties transforming knowledge into the appropriate form when merging disparate knowledge bases for the same knowledge representation system.)

Despite these problems, the development and use of a good knowledge interchange format is a worthwhile activity. It allows us to get multiple knowledge bases into a single language; and it allows us to write axioms relating the concepts used in those knowledge bases. As a result, we can begin practical knowledge interchange today – by reading the knowledge bases ourselves, by semi-automatically transforming them into more useful forms, by submitting the axioms to general reasoning programs to create new programs with the same functionality as the originals (though perhaps with less efficiency). And, of course, there is value for tomorrow as well. The existence of difficult but clearly stated knowledge transformation problems will provide the data to help researchers design newer and better knowledge interchange technology.

The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases

Thomas R. Gruber
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Building C
Palo Alto, CA 94304
gruber@sumex-aim.stanford.edu

Research and development in AI is impeded by an inability to share and reuse knowledge bases. The expense of building serious knowledge bases and the lack of means to exchange them with colleagues makes it difficult to generate, evaluate, and build on research results which depend on "domain theories" or "background knowledge." Similarly, commercial systems development is hampered by the lack of interoperability among tools based on different knowledge representations, and the lack of mechanisms for accumulating libraries of reusable knowledge.

Today's knowledge systems are isolated monoliths characterized by *high internal coupling* (e.g., among ground domain facts, procedures, terminology, axioms, and idiosyncratic ways of modelling the world) and a *lack of external coupling interfaces* that would enable the developer to reuse software tools and knowledge bases as modular components. The only practical way to "share" or "reuse" an existing knowledge base is to adopt the entire representation and programming environment of the existing KB.

What can be done to enable the accumulation, sharing, and reuse of knowledge bases? A proven software engineering approach is to *decompose* the monolithic system into reusable building blocks at modular boundaries. Three important decomposition techniques are already found in the AI methodology for building software systems:

- Separate knowledge from programs with a declarative knowledge representation language.
- Identify general classes and relations underlying application-specific facts, and organize knowledge to enable inheritance from these constructs.
- Characterize general problem solving tasks (e.g., classification) and classes of inference (e.g., subsumption), and design corresponding methods and algorithms.

Although they help, these techniques are insufficient to support sharability because the problem remains underconstrained. There are many ways to formalize

declarative knowledge, organize class and relation hierarchies, and characterize tasks and inferences. To achieve sharable, reusable knowledge bases we also need to:

- Specify a **canonical form** for declarative knowledge: a representation language with a standard syntax and semantics (for operators such as $\wedge, \vee, \neg, \Rightarrow, \exists$). A proposal for such a language is described elsewhere [Genesereth, 1991].
- Define **common ontologies**: vocabularies of representational terms—classes, relations, functions, object constants—with agreed-upon *definitions*, in the form of human readable text and machine-enforceable, declarative constraints on their well-formed use. Definitions may include restrictions on domains and ranges, placement in subsumption hierarchies, class-wide facts inherited to instances, and other axioms.

The purpose of this strategy is to enable the "literary" publication and exchange of formally-represented knowledge.¹ The canonical form provides the linguistic foundation, drawing a level boundary that factors out differences in *notation* among knowledge bases. Ontologies capture reusable intellectual *content* of a representation effort—the choices about classes and relations that are potentially relevant for describing a domain or performing a task. Each ontology embodies a set of ontological commitments in a form that enables one to build knowledge bases and tools based on those same commitments. Given a common language and vocabulary, one can build knowledge bases that instantiate and specialize the shared classes and relations. The instantiations and specializations of the ontologies carry application-specific information. Thus, the role of ontologies is to specify a *modular coupling* among bodies of knowledge and the tools that operate on them, serving as knowledge-level protocols for input, output, and communication.

¹Recent activities along these lines are described in [Gruber, 1990] and [Neches, 1991].

The aim is to build *libraries* of shared, reusable knowledge. If the specification of a standard declarative language is like a grammar of English, ontologies are reference works akin to dictionaries. Libraries could contain “off-the-shelf” knowledge-based tools that perform well-defined tasks such as varieties of simulation, diagnosis, etc. Ontologies specify the terms by which a tool user writes the “domain knowledge” for the tool, such as the equation models that drive a simulation or the components and failure mode descriptions used by the diagnostic engine. A knowledge library could also contain reusable fragments of domain knowledge, such as component models (e.g., of transistors, gears, valves) that can be composed to produce device models (e.g., of amplifiers, servo mechanisms, and hydraulic systems). Ontologies define various ways of modelling electrical, mechanical, and fluid flow mechanisms that make such reusable component libraries possible.

Like reusable software, knowledge bases have to be designed for sharability, organized by ontologies designed to support reuse. Mechanical translation of existing knowledge bases is no panacea; the reusable content of a knowledge base must be teased out and formulated to minimize hidden assumptions. The defined terms in ontologies identify and deliver the product of the representation effort, which is reused when the terms are instantiated in new applications. Note that the leverage does not come from reusing entire knowledge bases, filled with millions of ground facts about a broad domain of discourse (c.f. [Lenat and Guha, 1990]). Knowledge-based systems will always require application-specific knowledge, and no effort could hope to anticipate all possible content. The leverage comes from applying the ontologies that embody the representational choices underlying such knowledge bases in the design and construction of similar knowledge bases for different institutions and domains. Making such ontologies explicit and public can create a market for tools designed to operate on domain knowledge represented using a well-defined, agreed-upon vocabulary.

What are the technical problems in designing a common ontology? Space permits only an incomplete enumeration.

- What information about terms is most critical for supporting sharability? The names? Textual definitions? Type, arity, and argument restrictions? Arbitrary axioms?
- How can we achieve group consensus on “what to represent” when participating researchers have commitments to different tasks, representation tools, and domains? How can one describe assumptions and purposes of a particular ontology?
- How can one capture and use design rationale for representation (ontology) design?
- What kinds of automated assistance can support

the development of consistent sets of terms in a collaborative setting?

- What mechanisms can be used to verify compatibility with an ontology?
- How does one write correspondence theories that relate different ontologies (e.g., multiple ways of modelling device behavior)?

To investigate these and other questions we conducted a pilot study in the collaborative development of ontologies. In the Summer Ontology Project [Gruber, in preparation], computer scientists and engineers from several groups at Stanford and Bay Area research labs met to develop common ontologies for describing a class of electromechanical systems. We focused on four existing “motion control” devices, attempting to model their physical structure and dynamic behavior with several languages and approaches. From an analysis of these models and many hours of group discussion we have begun to define a set of representational terms. To accommodate the differing representation styles of the participants, we developed a translation mechanism called *Ontolingua* by which term definitions and axioms in the standard interchange representation can be automatically transformed into the corresponding forms for implemented frame systems and predicate calculus languages. The primary outcome to date is an ontology for lumped-parameter behavior models of physical devices that encompasses several modelling languages used in AI and Engineering. As a side product we defined an ontology for class-instance/slot representations: a portable vocabulary that unifies notions of classes and instances, relations, slots, and metaslots. When complete these ontologies and others under development will be distributed for evaluation and experimental use.

References

- Genesereth, M. R. (1990). DSL Reference Manual. Stanford University, Computer Science, Report Logic-90-3.
- Genesereth, M. R. (1991). Knowledge Interchange Format, Version 2.1 Reference Manual. Stanford University, Computer Science, Report Logic-90-5.
- Gruber, T. R. (1990). The Development of Large, Shared Knowledge-Bases: Collaborative Activities at Stanford. Stanford Knowledge Systems Laboratory, Technical Report KSL 90-62.
- Gruber, T. R. (in preparation). An Experiment in the Collaborative Development of Shared Ontology. Stanford Knowledge Systems Laboratory.
- Lenat, D. B. and Guha, R. V. (1990). *Building Large Knowledge-based Systems*. Addison-Wesley.
- Neches, R. (1991). Long-term Goals for Knowledge Representation Systems. *AI Magazine*, forthcoming.

Subject Index

A

abduction 70, 429, 535
abductive reasoning 70
action 250, 376
actions 313
 simultaneous 3
Adams' ε -semantics 175
adjunct 558
agents
 behavior specifications of 238
 informal 238
 multiple 301
 rational 473
AGM postulates 387
algorithm
 belief revision 99
 update 99
amalgamation 22
ampliative inference 526
analogical reasoning 139
analogy 139
anytime reasoning 250
argument systems 526
aspects 558
assertional default theories 496
assumption persistence 3
assumption-based reasoning 485
ATMS's 262
attachment 405
 procedural 405
 semantic 405
 theorem proving 405
 universal 405
attitudes, propositional 507
autoepistemic logic 230, 395
automata theory 128
axiomatization 269

B

bandwidth- k graph 186
base revision 417
BDI-architecture 473
behavior specification of agents 238
belief revision 99, 163, 387, 417, 570
belief revision algorithm 99
bilattices 250

C

calculus
 sequent 78
 situation 376
causal
 explanation 441, 535
 independence 441
 inference 15
causality 313
causation 441
change 376
 minimal 269
 theory 269
choice inference 50
circumscription 289, 347, 376
classification 128, 151
 taxonomic 335
clique tree 186
commitment 473
commonsense reasoning 15, 50
complement 558
completeness 186
completeness 269, 514
complexity of reasoning 70, 99, 151, 325, 335, 570
computational complexity 417
concept 78
concept languages 151, 335
conditional 357
 knowledge bases 269
 logic 50, 269, 357
 object 175
conflict 289
consistency 22, 186
constraint
 logic 196, 429
 monotonicity 547
 network 186
 propagation 111
 satisfaction problem 186
contexts, epistemic 507
contraction 78, 387
control of reasoning 262
conventions, narrative 15
coupling 601
credulous
 inference 50
 reasoning 208
cumulative default logic 496
cumulativity 496
cut 175

D

decision procedures 514
declarative past 40
decomposability, serial 89
deduction 196
default 496
 logic 118, 230, 496
 reasoning 289, 417, 453, 496
defeasible
 argumentation 526
 reasoning 118, 526
defensible reasoning 50
diagnosis 70
diagonal lemma 507
disjunctive
 databases 230
 reasoning 230

E

entailment, pragmatic 50
epistemic
 contexts 507
 entrenchment 417
 inconsistency 453
 logic 325
 relevance 417
equality 196
events 3
evidential reasoning 535
executable temporal logic 40
expansion 395
explaining away 535
explanation 70
 causal 441, 535
explicit belief 368
extension 230

F

first-order logic 22, 405
formal semantics 208
frame
 axiom 262
 problem 250, 376
functions, multivariate 547
future, imperative 40

G

Gärdenfors postulates 417
generalization 429
generic induction 526

graphs, update 89
groups, permutation 89

H

hierarchical planning 262
hierarchy
 inheritance 61
 ISA 128
 metalinguistic 507
hybrid
 reasoning 196, 405, 514
 representation systems 405
hypothetical reasoning 61

I

imperative future 40
implemented systems 591
implicit belief 368
incomplete information 238
inconsistent default logic 453
incremental program development 238
independence, causal 441
indexical knowledge 301
induction 429
 generic 526
 statistically based 526
inductive
 learning 441
 reasoning 441
inference
 ampliative 526
 causal 15
 choice 50
 credulous 50
 nonmonotonic 32
 skeptical 50
informal agents 238
inheritance 289
 hierarchy 61
intension 558
intention
 logic 473
 revision 473
intercausal reasoning 535
interpreter, temporal logic 40
intuitionistic logic 22

J

justification 15

K

KL-ONE 15, 335
knowledge
 bases, shared 601
 interchange 601
 representation 78
k-trees 186

L

language, Meta 40
languages concept 151
learning 429
 inductive 441
 model 89, 429
 pac 429
lemma, diagonal 507
limited rationality 163
linear logic 78
linear-time temporal logic 40
logic
 conditional 50, 269, 357
 constraint 196, 429
 cumulative default 496
 default 118, 230, 496
 epistemic 325
 executable temporal 40
 first-order 22, 405
 inconsistent default 453
 intention 473
 linear 78
 linear-time temporal 40
 modal 196, 507
 model 357
 non classical 453
 nonmonotonic 15, 32, 50, 357
 ordered 208
 paraconsistent 453
 pragmatic 50
 programming 40, 61, 485, 570
 relevance 368
 sorted 429
 temporal 111
 executable 40
 interpreter 40
three-valued 570
update 269
logical omniscience 325, 368
logics for naming 301

M

macro operators 89
maintenance
 reason 570
 truth 163, 485
mathematical induction 347
Meta
 language 40
 reasoning 61
metalevel 22
 reasoning 40, 262
metalinguistic hierarchy 507
metalogue 61
metaphor 139
Meta-programming 40
MetateM 40
methodology 591
minimal
 change 269, 387
 models 441
minimality 186
modal
 logic 22, 196, 507
 first-order 301
 S5 301
 operators 250
 predicates 507
 modality, syntactical 507
model
 checking 325
 learning 89
models 289
 of conditionals 269
 of updates 269
 stable 485
monotonicity constraints 547
Montague's proof 507
multiple
 agents 301
 extension problem 289
 scenarios 526
multi-theory reasoning 61
multivariate functions 547

N

naming 301
narrative conventions 15
natural deduction 22
network
 neural 32
 semantic 558
networks, probabilistic 441
neural network 32
Newton's first law 347

non classical logic 453
non contradiction principle 453
nonmonotonic
 inference 32
 logic 15, 32, 50, 357, 395
 reasoning 118, 163, 175, 208,
 262, 289, 376, 453, 485, 496
non-standard logics 368

O

object orientation 61
Occam's razor 441
ontology 558, 601
operators, macro 89
ordered logic 208
orientation, object 61

P

paraconsistency 453
paraconsistent logic 453
paramodulation 196
partial programming 238
past, declarative 40
permutation groups 89
persistence assumption 3
plan abstraction 128
planning 3, 313, 473
 hierarchical 262
plans 128
possible
 models 387
 worlds 196, 269, 301, 325, 368, 387, 473, 526
possibility
 theory 175
 worlds 196, 269, 325
pragmatic
 entailment 50
 logic 50
 reasoning 50
predicates, modal 507
preferences 289
preferential
 entailment 357
 model 357
principle, non contradiction 453
prioritization 289
probabilistic
 causality 441
 reasoning 535
probability 313
 theory 175
problem
 constraint satisfaction 186
 frame 250

multiple extension 289
ramification 376
reformulation 514
side-effect 473
synthesis 262
Yale shooting 15, 250, 453, 526
procedural hints 238
procedures, decision 514
program development
 development 238
 semantics 238
programming
 logic 40, 61, 485
 meta- 40
 partial 238
projection problem, temporal 15
proof 78
 Montague's 507
proposition 558
propositional attitudes 507
proving, theorem 514

Q

QSIM 547
qualification axiom n 262
qualitative
 reasoning 535, 547
 synergy 535, 547
quotation terms 507

R

ramification problem 376
rational
 agents 473
 inference 163
ravens 118
reasoning
 abductive 70
 about actions 238
 about change 99
 anytime 250
 assumption-based 485
 commonsense 15, 50
 complexity of 70, 99, 151, 325, 335, 570
 control of 262
 default 289, 496
 defeasible 50, 118, 526
 defensible 50
 disjunctive 230
 evidential 535
 hybrid 514
 hypothetical 61
 inductive 441

intercausal 535
meta 61
meta-level 40
multi-theory 61
nonmonotonic 118, 175, 208,
262, 289, 376, 453, 485, 496
pragmatic 50
probabilistic 535
qualitative 535, 547
semantic 325
taxonomic 151, 335
temporal 3, 15, 111, 250
reason maintenance 163, 570
recursive semantics 453
reflection 22
reformulation, problem 514
relative names 301
relevance logic 368
representation, knowledge 78
resolution 196
restricted monotonicity 175
revision
 belief 99, 163, 387, 417, 570
 belief algorithm 99
 intention 473

S

satisfiability 151
scenarios, multiple 526
semantic
 network 558
 reasoning 325
semantics 496
 formal 208
 logic programming 570
 of programs 238
 recursive 453
sequent calculus 78
serial decomposability 89
side-effect problem 473
simultaneous actions 3
situation 558
 calculus 376
skeptical inference 50
social choice theory 163
sorted logic 429
spatial reasoning 347
specificity 289
Spohn's conditional functions 175
stable models 485
statistically based induction 526
story understanding 15
subnormal modal logics 395
substitutional framework 196
subsumption 78, 128, 151, 335
subsymbolic 32

synergy, qualitative 535, 547
syntactical modality 507
synthesis problems 262
systems
 argument 526
 implemented 591

T

taxonomic
 classification 335
 reasoning 151, 335
taxonomy 601
temporal
 linear-time logic 40
 logic 111, 313
 logic, executable 40
 projection problem 15
 reasoning 3, 15, 111, 250
terminological
 reasoning 151
 systems 128, 601
terms, quotation 507
theorem proving 325, 405, 514
theories, assertional default 496
theory
 change 269, 387
 resolution 405
three-valued logic 570
time 3
tolerance spaces 347
truth maintenance 163, 485

U

understanding, story 15
update 387
 algorithm 99
 graphs 89
 logic 269
updates 99
 models of 269

W

weakening 78
worlds, possible 196, 269, 301, 473, 526

Y

Yale shooting problem 15, 250, 453, 526

Author Index

Allen, James F. 3
Amsterdam, Jonathan 15
Attardi, Giuseppe 22, 597

Baader, Franz 335
Balkenius, Christian 32
Barringer, Howard 40
Bell, John 50
Brogi, Antonio 61
Buchsbaum, Arthur 453
Bylander, Tom 70

Carnes, James R. 111
Castaing, Jacqueline 78
Chalasani, Prasad 89
Chester, Daniel L. 139
Chou, Timothy S-C 99

Davis, William S. 111
Delgrande, James P. 118
Devanbu, Premkumar T. 128
Dierbach, Charles 139
Donini, Francesco M. 151
Doyle, Jon 163
Dubois, Didier 175

Etzioni, Oren 89

Fisher, Michael 40
Freuder, Eugene C. 186
Frisch, Alan M. 196, 429

Gabbay, Dov 40, 208
Gärdenfors, Peter 32, 585
Geffner, Hector 218
Gelfond, Michael 230
Genesereth, Michael R. 238, 599
Georgeff, Michael P. 473
Ginsberg, Matthew L. 250, 262
Grahne, Gösta 269
Greiner, Russell 277
Grosof, Benjamin N. 289
Grove, Adam J. 301
Gruber, Thomas R. 601

Haddawy, Peter 313
Halpern, Joseph Y. 301, 325
Henrion, Max 535
Hollunder, Bernhard 335
Hsu, Jane Yung-jen 238
Hunter, Anthony 40

Jackson, W. Ken 118

Katsuno, Hirofumi 387
Kaufman, Stephen G. 347
Laenen, E. 208
Lakerneyer, Gerhard 368
Lamarre, Philippe 357
Lenzerini, Maurizio 151
Lifschitz, Vladimir 230, 376
Litman, Diane J. 128

Marek, Wiktor 395
Mark, William 595
Mendelzon, Alberto O. 387
Mount, John 89
Myers, Karen L. 405

Nardi, Daniele 151
Nebel, Bernhard 417
Nutt, Werner 151

Orponen, Pekka 277

Page, C. David, Jr. 429
Pearl, Judea 441
Pequeno, Tarcisio 453
Pimentel, Stephen G. 485
Poole, David 461
Prade, Henri 175
Provan, Gregory M. 461
Przymusinska, Halina 230

Rao, Anand S. 473
Rich, Charles 591
Rodi, William L. 485

Schaub, Torsten 496
Scherl, Richard B. 196
Schweizer, Paul 507
Shvarts, Grigori F. 395
Simi, Maria 22

Truszczynski, Miroslaw 230, 395
Turini, Franco 61

Van Baalen, Jeffrey 514
Vardi, Moshe Y. 325
Verma, T. S. 441
Vermeir, D. 208
Vreeswijk, Gerard 526

Wellman, Michael P. 535, 547
Wilensky, Robert 558
Winslett, Marianne 99
Witteveen, Cees 570

Q 387 .P75 1991 C.1
Principles of knowledge repres
Stanford University Libraries



3 6105 031 381 101

Q387
IS9
1991

DATE DUE

[REDACTED]	APR 1 8 1997
[REDACTED]	JUN 19 1997
[REDACTED]	JUN 05 2000
[REDACTED]	NOV 01 2000
[REDACTED]	SEP 15 1995
[REDACTED]	JAN 03 2001
[REDACTED]	MAR 08 2005
[REDACTED]	JUN 15 2005
[REDACTED]	MAY 03 2006
[REDACTED]	GRAD 06 1996
	PRINTED IN U.S.A.

STANFORD UNIVERSITY LIBRARIES
STANFORD, CALIFORNIA 94305-6004

PRINCIPLES OF KNOWLEDGE REPRESENTATION AND REASONING: PROCEEDINGS OF THE SECOND INTERNATIONAL CONFERENCE

Edited by James Allen (University of Rochester), Richard Fikes (Stanford University) and Erik Sandewall (Linköping University)

The idea of explicit representations of knowledge, manipulated by general-purpose inference algorithms, underlies much work in artificial intelligence, from natural language processing to expert systems. The papers included in these proceedings focus on the principles of automated representation and reasoning, as distinct from details of implementation.

Each contribution was written by a leading scientist in the field, and topics covered include the logical specifications of reasoning behaviors and representation formalisms, comparative analysis of competing algorithms and formalisms, and analyses of the correctness and computational complexity of reasoning algorithms. These proceedings are an essential reference volume for researchers and students interested in a detailed view of this key research area.

Additional titles of interest from the Morgan Kaufmann Series in Representation and Reasoning

REPRESENTATIONS OF COMMONSENSE KNOWLEDGE, Ernest Davis (Courant Institute of Mathematical Sciences, NYU)

REASONING ABOUT PLANS, James Allen (University of Rochester), Henry Kautz (AT&T Bell Laboratories), Richard Pelavin (Philips Laboratories) and Josh Tenenberg (University of Rochester)

PRINCIPLES OF SEMANTIC NETWORKS: EXPLORATIONS IN THE REPRESENTATION OF KNOWLEDGE, edited by John Sowa (IBM Systems Research)

READINGS IN PLANNING, edited by James Allen (University of Rochester), James Hendler (University of Maryland) and Austin Tate (Edinburgh University)

ISBN 1-55860-165-1
ISSN 1046-9567
Artificial Intelligence

Morgan Kaufmann Publishers
2929 Campus Drive, Suite 260
San Mateo, CA 94403

