Thomas Lukasiewicz
Attila Sali (Eds.)

LNCS 7153

Foundations of Information and Knowledge Systems

7th International Symposium, FoIKS 2012
Kiel, Germany, March 2012
Proceedings

Springer

Lecture Notes in Computer Science 7153

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Alfred Kobsa
University of California, Irvine, CA, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
TU Dortmund University, Germany

Madhu Sudan
Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Gerhard Weikum
Max Planck Institute for Informatics, Saarbruecken, Germany

oarieli@mta.ac.il

Thomas Lukasiewicz Attila Sali (Eds.)

Foundations of Information and Knowledge Systems

7th International Symposium, FoIKS 2012
Kiel, Germany, March 5-9, 2012
Proceedings

Springer

oarieli@mta.ac.il

Volume Editors

Thomas Lukasiewicz
University of Oxford
Department of Computer Science
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
E-mail: thomas.lukasiewicz@cs.ox.ac.uk

Attila Sali
Hungarian Academy of Sciences
Alfréd Rényi Institute
Reáltanoda utca 13-15, 1053 Budapest, Hungary
E-mail: sali.attila@renyi.mta.hu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-28471-7 e-ISBN 978-3-642-28472-4
DOI 10.1007/978-3-642-28472-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012931445

CR Subject Classification (1998): G.2, F.4.1, I.2.3, I.2.4, D.3

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

© Springer-Verlag Berlin Heidelberg 2012
This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.
The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

oarieli@mta.ac.il

Preface

This volume contains the articles that were presented at the 7th International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2012), which was held in Kiel, Germany, March 5–9, 2012.

The FoIKS symposia provide a biennial forum for presenting and discussing theoretical and applied research on information and knowledge systems. The goal is to bring together researchers with an interest in this subject, share research experiences, promote collaboration, and identify new issues and directions for future research.

FoIKS 2012 solicited original contributions on foundational aspects of information and knowledge systems. This included submissions that apply ideas, theories or methods from specific disciplines to information and knowledge systems. Examples of such disciplines are discrete mathematics, logic and algebra, model theory, information theory, complexity theory, algorithmics and computation, statistics, and optimization.

Previous FoIKS symposia were held in Sofia (Bulgaria) in 2010, Pisa (Italy) in 2008, Budapest (Hungary) in 2006, Vienna (Austria) in 2004, Schloss Salzau near Kiel (Germany) in 2002, and Burg/Spreewald near Berlin (Germany) in 2000. FoIKS took up the tradition of the conference series Mathematical Fundamentals of Database Systems (MFDBS), which initiated East–West collaboration in the field of database theory. Former MFDBS conferences were held in Rostock (Germany) in 1991, Visegrad (Hungary) in 1989, and Dresden (Germany) in 1987.

The FoIKS symposia are a forum for intense discussions. Speakers are given sufficient time to present their ideas and results within the larger context of their research. Furthermore, participants are asked in advance to prepare a first response to a contribution of another author in order to initiate discussion.

Suggested topics for FoIKS 2012 included, but were not limited to:

- Database Design: formal models, dependencies and independencies
- Dynamics of Information: models of transactions, concurrency control, updates, consistency preservation, belief revision
- Information Fusion: heterogeneity, views, schema dominance, multiple source information merging, reasoning under inconsistency
- Integrity and Constraint Management: verification, validation, consistent query answering, information cleaning
- Intelligent Agents: multi-agent systems, autonomous agents, foundations of software agents, cooperative agents, formal models of interactions, logical models of emotions
- Knowledge Discovery and Information Retrieval: machine learning, data mining, formal concept analysis and association rules, text mining, information extraction

oarieli@mta.ac.il

- Knowledge Representation, Reasoning and Planning: non-monotonic formalisms, probabilistic and non-probabilistic models of uncertainty, graphical models and independence, similarity-based reasoning, preference modeling and handling, argumentation systems
- Logics in Databases and AI: classical and non-classical logics, logic programming, description logics, spatial and temporal logics, probability logic, fuzzy logic
- Mathematical Foundations: discrete structures and algorithms, graphs, grammars, automata, abstract machines, finite model theory, information theory, coding theory, complexity theory, randomness
- Security in Information and Knowledge Systems: identity theft, privacy, trust, intrusion detection, access control, inference control, secure Web services, secure Semantic Web, risk management
- Semi-Structured Data and XML: data modeling, data processing, data compression, data exchange
- Social Computing: collective intelligence and self-organizing knowledge, collaborative filtering, computational social choice, Boolean games, coalition formation, reputation systems
- The Semantic Web and Knowledge Management: languages, ontologies, agents, adaption, intelligent algorithms
- The WWW: models of Web databases, Web dynamics, Web services, Web transactions and negotiations

This volume contains 20 articles (12 regular and 8 short presentations), which were selected out of 53 submissions for FoIKS 2012 in a rigorous reviewing process, where each submission was reviewed by at least three Program Committee members. The volume also contains articles for two of the three invited talks.

We wish to thank all authors who submitted papers and all conference participants for fruitful discussions. We are grateful to András Benczúr, Thomas Eiter, and Thomas Schwentick for their invited talks. We would like to thank the Program Committee members and additional reviewers for their timely expertise in carefully reviewing the submissions. Special thanks to Bernhard Thalheim and his team (in particular, Stefanie Jureit, Steffen Gaede, Kai Jannaschk, Hans-Joachim Klein, René Noack, and Ove Sörensen) from the Christian Albrechts University Kiel for the organization of the conference and wonderful days in Kiel. We want to thank Markus Kirchberg for his work as Publicity Chair. Many thanks also to the developers of the EasyChair Conference System, which we used for the reviewing process and the preparation of this volume.

March 2012

Thomas Lukasiewicz
Attila Sali

oarieli@mta.ac.il

Conference Organization

FoIKS 2012 was organized by the Christian Albrechts University Kiel, Germany.

Program Chairs

Thomas Lukasiewicz	University of Oxford, UK
Attila Sali	Alfréd Rényi Institute, Hungarian Academy of Sciences, Hungary

Local Organization Chair

Bernhard Thalheim	Christian Albrechts University Kiel, Germany

Publicity Chair

Markus Kirchberg	Hewlett-Packard Labs, Singapore

Program Committee

José Júlio Alferes	Universidade Nova de Lisboa, Portugal
Leila Amgoud	University of Toulouse, France
Paolo Atzeni	Roma Tre University, Italy
Salem Benferhat	University of Lens, France
Leopoldo Bertossi	Carleton University, Canada
Philippe Besnard	University of Toulouse, France
Joachim Biskup	University of Dortmund, Germany
Piero A. Bonatti	University of Naples "Federico II", Italy
Andrea Calì	Birkbeck, University of London, UK
Paolo Cappellari	Dublin City University, Ireland
Jan Chomicki	University at Buffalo, USA
Alfredo Cuzzocrea	ICAR-CNR and University of Calabria, Italy
Marina De Vos	University of Bath, UK
Michael I. Dekhtyar	Tver State University, Russia
James P. Delgrande	Simon Fraser University, Canada
Tommaso Di Noia	Technical University of Bari, Italy
Jürgen Dix	Clausthal University of Technology, Germany
Francesco M. Donini	University of Tuscia, Italy
Thomas Eiter	Vienna University of Technology, Austria
Ronald Fagin	IBM Almaden Research Center, San Jose, USA
Victor Felea	"Al.I. Cuza" University of Iaşi, Romania
Flavio Ferrarotti	Victoria University of Wellington, New Zealand

oarieli@mta.ac.il

Sergio Flesca	University of Calabria, Italy
Lluis Godo	Artificial Intelligence Research Institute (IIIA - CSIC), Spain
Gianluigi Greco	University of Calabria, Italy
Claudio Gutierrez	University of Chile, Chile
Sven Hartmann	Clausthal University of Technology, Germany
Stephen J. Hegner	Umeå University, Sweden
Edward Hermann Haeusler	Pontifícia Universidade Católica do Rio de Janeiro, Brazil
Andreas Herzig	University of Toulouse, France
Pascal Hitzler	Wright State University, USA
Eyke Hüllermeier	University of Marburg, Germany
Anthony Hunter	University College London, UK
Yasunori Ishihara	Osaka University, Japan
Gyula O.H. Katona	Alfréd Rényi Institute, Hungarian Academy of Sciences, Hungary
Gabriele Kern-Isberner	University of Dortmund, Germany
Attila Kiss	Eötvös Loránd University, Budapest, Hungary
Sébastien Konieczny	University of Lens, France
Gerhard Lakemeyer	RWTH Aachen, Germany
Jérôme Lang	University of Paris 9, France
Domenico Lembo	Sapienza University of Rome, Italy
Mark Levene	Birkbeck, University of London, UK
Sebastian Link	Victoria University of Wellington, New Zealand
Weiru Liu	Queen's University Belfast, UK
Sebastian Maneth	NICTA and University of New South Wales, Australia
Pierre Marquis	University of Artois, France
Wolfgang May	University of Göttingen, Germany
Carlo Meghini	ISTI-CNR Pisa, Italy
Leora Morgenstern	New York University, USA
Amedeo Napoli	LORIA Nancy, France
Wilfred S.H. Ng	Hong Kong University of Science and Technology, Hong Kong
Dan Olteanu	University of Oxford, UK
Henri Prade	University of Toulouse, France
Andrea Pugliese	University of Calabria, Italy
Sebastian Rudolph	University of Karlsruhe, Germany
Francesco Scarcello	University of Calabria, Italy
Klaus-Dieter Schewe	Software Competence Center Hagenberg, Austria
Dietmar Seipel	University of Würzburg, Germany
Nematollaah Shiri	Concordia University, Montreal, Canada
Gerardo I. Simari	University of Oxford, UK
Guillermo R. Simari	Universidad Nacional del Sur, Argentina
Nicolas Spyratos	University of Paris-South, France

oarieli@mta.ac.il

Umberto Straccia	ISTI-CNR Pisa, Italy
Letizia Tanca	Politecnico di Milano, Italy
Alex Thomo	University of Victoria, Canada
Krisztián Tichler	Eötvös Loránd University, Budapest, Hungary
Miroslaw Truszczynski	University of Kentucky, USA
José Maria Turull-Torres	Massey University Wellington, New Zealand
Wiebe van der Hoek	University of Liverpool, UK
Dirk Van Gucht	Indiana University, USA
Victor Vianu	University of California San Diego, USA
Evgenii E. Vityaev	Russian Academy of Sciences, Russia
Peter Vojtáš	Charles University, Czech Republic
Jef Wijsen	University of Mons-Hainaut, Belgium
Stefan Woltran	Vienna University of Technology, Austria

Additional Reviewers

David Carral Martínez	Long Chen	Sean Chester
Anna Corazza	Minh Dao-Tran	Michel de Rougemont
Wolfgang Dvorák	Ingo Feinerer	Alejandro Leonardo Grosso
Maria Vanina Martinez	João Moura Pires	Carles Noguera
Luigi Sauro	Vadim Savenkov	Edoardo Serra
Maryam Shoaran	Cristina Sirangelo	Tony Tan
Thu Trinh	Mars K. Valiev	

Local Organization

Stefanie Jureit
Steffen Gaede
Kai Jannaschk
Hans-Joachim Klein
René Noack
Ove Sörensen

oarieli@mta.ac.il

Table of Contents

Invited Talks

Uniform Evaluation of Nonmonotonic DL-Programs 1
Thomas Eiter, Thomas Krennwallner, Patrik Schneider, and Guohui Xiao

Foundations of XML Based on Logic and Automata: A Snapshot 23
Thomas Schwentick

Regular Articles

Inconsistency-Tolerance in Knowledge-Based Systems by Dissimilarities 34
Ofer Arieli and Anna Zamansky

Revising Belief without Revealing Secrets 51
Joachim Biskup and Cornelia Tadros

More Than the Sum of Its Parts – Holistic Ontology Alignment by Population-Based Optimisation 71
Jürgen Bock, Sebastian Rudolph, and Michael Mutter

Handling Preferences in P2P Systems 91
Luciano Caroprese and Ester Zumpano

Backing and Undercutting in Abstract Argumentation Frameworks 107
Andrea Cohen, Alejandro J. García, and Guillermo R. Simari

The Impact of Transitive Closure on the Boolean Expressiveness of Navigational Query Languages on Graphs 124
George H.L. Fletcher, Marc Gyssens, Dirk Leinders, Jan Van den Bussche, Dirk Van Gucht, Stijn Vansummeren, and Yuqing Wu

Using Functional Dependencies for Reducing the Size of a Data Cube . . . 144
Eve Garnaud, Sofian Maabout, and Mohamed Mosbah

Armstrong Databases and Reasoning for Functional Dependencies and Cardinality Constraints over Partial Bags 164
Sven Hartmann, Henning Köhler, Sebastian Link, and Bernhard Thalheim

FD Covers and Universal Complements of Simple Projections 184
Stephen J. Hegner

oarieli@mta.ac.il

Encoding Databases Satisfying a Given Set of Dependencies 203
Gyula O.H. Katona and Krisztián Tichler

On Lifted Inference for a Relational Probabilistic Conditional Logic with Maximum Entropy Semantics 224
Annika Krämer and Christoph Beierle

Flexible and Efficient Distributed Resolution of Large Entities 244
András J. Molnár, András A. Benczúr, and Csaba István Sidló

On Real-Valued Evaluation of Propositional Formulas 264
Aleksandar Perović, Dragan Doder, and Zoran Ognjanović

Detecting Suspect Answers in the Presence of Inconsistent Information 278
Olivier Pivert and Henri Prade

Learning the News in Social Networks 298
Krishnan Rajagopalan, Venkatesh Srinivasan, and Alex Thomo

Verifying Resource Requirements for Ontology-Driven Rule-Based Agents 312
Abdur Rakib, Rokan Uddin Faruqui, and Wendy MacCaull

Formalizing Production Systems with Rule-Based Ontologies 332
Martín Rezk and Michael Kifer

Count Constraints and the Inverse OLAP Problem: Definition, Complexity and a Step toward Aggregate Data Exchange 352
Domenico Saccà, Edoardo Serra, and Antonella Guzzo

Synchronous Parallel Database Transformations 370
Klaus-Dieter Schewe and Qing Wang

Functional Dependencies on Extended Relations Defined by Regular Languages 384
Gyula I. Szabó and András Benczúr

Author Index 405

oarieli@mta.ac.il

Uniform Evaluation of Nonmonotonic DL-Programs*

Thomas Eiter, Thomas Krennwallner, Patrik Schneider, and Guohui Xiao

Institut für Informationssysteme, Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter,tkren,patrik,xiao}@kr.tuwien.ac.at

Abstract. Nonmonotonic description logic programs are a major formalism for a loose coupling of rules and ontologies, formalized in logic programming and description logics, respectively. While this approach is attractive for combining systems, the impedance mismatch between different reasoning engines and the API-style interfacing are an obstacle to efficient evaluation of dl-programs in general. Uniform evaluation circumvents this by transforming programs into a single formalism, which can be evaluated on a single reasoning engine. In this paper, we consider recent and ongoing work on this approach which uses relational first-order logic (and thus relational database engines) and datalog with negation as target formalisms. Experimental data show that significant performance gains are possible and suggest the potential of this approach.

1 Introduction

In the past decade, the growing importance of the Web and its envisioned future development has triggered a lot of research on accessing and processing data based on semantic approaches. The distributed nature of the Web poses a challenge for semantic integration, even at the level of weak interoperability of different sites. To mitigate this problem, standard knowledge representation formats have been conceived in the layered architecture of the so called Semantic Web, in which the Web Ontology language (OWL) and the more recent Rule Interchange Format (RIF) play a prominent role.

In this context, the issue of combining rules and ontologies has been considered in a number of works; see [12,41,15] for some recent surveys. Among several approaches, loose coupling of rules and ontologies is one which aims at combining respective knowledge bases by means of a clean interfacing semantics, in which roughly speaking inferences are mutually exchanged such that the one knowledge base takes the imported information into account, and exports in turn conclusions to the other knowledge base. This approach is fostered by nonmonotonic description logic (dl-) programs [17], where this exchange is handled by a generalization of the answer set semantics of nonmonotonic logic programs [23]. Follow up work has adapted this approach to other formalisms (e.g., [51,29,20]) and considered alternative semantics (e.g. [39,52,16]).

The loose coupling approach is attractive in several regards. First, legacy knowledge bases, powered by different reasoning engines, can be combined. Second, thanks to

* This work has been supported by the Austrian Science Fund (FWF) projects P20840 & P20841 and by the EC ICT Integrated Project Ontorule (FP7 231875).

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 1–22, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

the interfacing and loose semantics connection, it is fairly easy to incorporate further knowledge formats besides rules and OWL (description logic) ontologies, e.g. RDF knowledge base; HEX-programs [20] are a respective generalization of dl-programs, which in fact allow for incorporating arbitrary software. And third, view based data access of loose coupling is in support of privacy, as the internal structure of a knowledge base remains hidden.

On the other hand, the impedance mismatch of different formalisms and reasoning engines comes at a price. A simple realization of the loose coupling considers the interface calls as an API which makes computation expensive, in particular if rules lead to choices via the underlying semantics. The black box view of other knowledge bases hinders optimization and is a major obstacle for scalability.

There is a ray of hope, however, if information about the internal structure of an accessed knowledge base is available. The simplest way is to give up privacy and make the knowledge base transparent such that its axioms and semantics are known ("open source"). Other possibility is to not reveal all information, but some abstract properties [19]. This still, however, leaves the impedance mismatch between different reasoning engines.

To overcome the latter, a suggestive approach is to convert the evaluation problem into one for a single reasoning engine, which means to transform a dl-program into an (equivalent) knowledge base in one formalism for evaluation (ideally, in one already considered). This opens a middle ground for evaluation and privacy, as the transformation may hide or blur the internal structure of the knowledge base.[1]

This idea of a "uniform evaluation" approach raises several issues.

1. Naturally, the cost of a transformation, and whether such a transformation is efficiently possible. In a sense, efficiency means that the overall evaluation cost does not increase with respect to some measure (typically, worst case complexity). Here, notions of embedding of a formalism into another might be considered, and besides computational also semantic properties like modularity are of interest, cf. [31].
2. At a more foundational level, whether a transformation to a target formalism does exist after all, if resources for its computation are disregarded, or even allowed to be not computable. Here one may further consider whether the transformation is ad hoc, for concrete knowledge bases embraced with a dl-program, or whether it is independent of their data (factual resp. assertional) parts.
3. As for the evaluation, the complexity of the target formalism, where –as common in the study of data and knowledge representation formalisms– the data complexity (i.e., complexity under varying data) deserves particular attention. Transformation to a formalism with lower complexity comes inevitably at some cost, which usually means an exponential increase in the size of the knowledge base. Popular examples of this in Description Logics are first-order rewriting of conjunctive query answering over DL-Lite ontologies [8] or the reduction of $\mathcal{SHIQ}$ to disjunctive datalog [30].
4. The feasibility of transformations for practical concerns, in particular for evaluation using available technology and platforms. In this regard, it is of interest to see whether theoretical results, as obtained for the items 1 and 3, live up to practical

[1] Note, however, that uniform evaluation is different from tight integration of KBs in a single unifying logic, cf. [41].

oarieli@mta.ac.il

realization. For example, an exponential blowup of the DL-Lite rewriting in [8], which is prohibitive in some cases, can be mitigated [45] or avoided using other notions of rewritings [36].

As for items 1 and 2, embeddings of dl-programs into various well-known non-monotonic logics have been studied, among them Autoepistemic Logic [13], Equilibrium Logic [21] (a logic-based version of Answer Set Semantics), Reiter's Default Logic [53], and MKNF [41]; however, these works targeted more semantic aspects than evaluation.

In recent and ongoing works, uniform evaluation of various fragments of dl-programs has been considered at the KBS group of TU Wien; in particular, transformation to first-order logic [16] and to datalog with negation [28,55]. This paper reviews some of this work with a focus on items 1 and 4 from above, and reports some experimental data.

The main observation is that, expectedly, uniform evaluation leads to significant performance improvements compared to simple (standard) evaluation of dl-programs coupling different reasoning engines, as done by the dlvhex reasoner.[2] The results show that the approach has potential, although further work is needed to boost scalability.[3]

The remainder of this article is organized as follows. In the next section, we briefly recall dl-programs. After that, we consider in Section 3 transformation of dl-programs to first-order logic, which makes evaluation using relational database technology possible. This requires to exclude recursion from rules and to have ontologies that are first-order rewritable. Section 4 considers transformation to datalog with negation, which hosts recursion in rules and for encodings of ontologies; as discussed in Section 5, the program can be naturally expressed in a modular version of datalog. The final Section 6 provides a discussion and gives an outlook on possible future work.

2 DL-Programs

We recall nonmonotonic description logic programs (simply dl-programs) under the answer set [17] and the well-founded semantics [16]. They combine Description Logics and nonmonotonic logic programs in a loose coupling, under a strict semantic separation.

2.1 Description Logics

Description Logics (DLs) are a well-known family of KR formalisms based on fragments of *first-order (FO) logic*. The vocabulary of a DL consists of *individual*, *class*, and *role names*; *knowledge bases* (KB) consist of a *terminological box* (TBox), which contains axioms about relations between classes and roles, and an *assertional box* (ABox), which contains factual knowledge about individuals. For the purpose of this paper, we first recall DL-Lite [8,2], the logical underpinning of OWL 2 QL; later, we consider $\mathcal{EL}$ [3,4], the underpinning of OWL 2 EL, and $\mathcal{LDL}^+$, which is a DL strongly related to Datalog and OWL 2 RL (for all these OWL 2 profiles, see [40]).

[2] `http://www.kr.tuwien.ac.at/research/systems/dlvhex/`

[3] Implementation information, benchmark instances, and further details on all benchmarks is available at `http://www.kr.tuwien.ac.at/research/systems/drew/experiments.html`.

oarieli@mta.ac.il

DL-Lite. Consider a vocabulary of individual names $\mathbf{I}$, *atomic concepts* $\mathbf{C}$, and *atomic roles* $\mathbf{R}$. Then, for A and P being an atomic concept and atomic role, respectively, we define *basic* concepts B and *basic* roles R, *complex* concepts C and *complex* role expressions E as

$$\begin{array}{rcl} B & ::= & A \mid \exists R \\ R & ::= & P \mid P^- \end{array} \qquad \begin{array}{rcl} C & ::= & B \mid \neg B \\ E & ::= & R \mid \neg R \end{array}$$

where P^- is the inverse of P.

A DL-Lite$_R$ *knowledge base* is a pair $L = (\mathcal{T}, \mathcal{A})$ where the TBox $\mathcal{T}$ consists of a finite set of *inclusion assertions* of the form $B \sqsubseteq C$ and $R \sqsubseteq E$, and the ABox $\mathcal{A}$ is a finite set of *membership assertions* on atomic concepts and on atomic roles of the form $A(a)$ and $P(a, b)$, where a and b are individual names of $\mathbf{I}$.

The semantics of DL-Lite$_R$ is given in terms of FO interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a nonempty domain and $\cdot^{\mathcal{I}}$ an *interpretation function* such that $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for all $a \in \mathbf{I}$, $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for all $A \in \mathbf{C}$, $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ for all $P \in \mathbf{R}$, and

- $(P^-)^{\mathcal{I}} = \{(a_2, a_1) \mid (a_1, a_2) \in P^{\mathcal{I}}\}$;
- $(\exists R)^{\mathcal{I}} = \{a_1 \mid \text{ there exists some } a_2 \text{ such that } (a_1, a_2) \in R^{\mathcal{I}}\}$;
- $(\neg B)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus B^{\mathcal{I}}$; and
- $(\neg R)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ *satisfies* a concept inclusion $C_1 \sqsubseteq C_2$ (resp. role inclusion $E_1 \sqsubseteq E_2$), if ${C_1}^{\mathcal{I}} \subseteq {C_2}^{\mathcal{I}}$ (resp. ${E_1}^{\mathcal{I}} \subseteq {E_2}^{\mathcal{I}}$), and satisfies a TBox $\mathcal{T}$, if it satisfies each inclusion assertion in $\mathcal{T}$. Furthermore, $\mathcal{I}$ satisfies $C(a)$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and satisfies $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$, and $\mathcal{I}$ satisfies an ABox $\mathcal{A}$, if it satisfies each assertion in $\mathcal{A}$. Finally, $\mathcal{I}$ *satisfies* $L = (\mathcal{T}, \mathcal{A})$, if it satisfies $\mathcal{T}$ and $\mathcal{A}$. A KB L (resp. TBox $\mathcal{T}$) *logically implies* an assertion α, if all models of L (resp. $\mathcal{T}$) satisfy α. As usual, satisfaction and logical entailment are denoted with $\models$.

2.2 Description Logic Programs

Description logic programs (L, P) have rules similar as logic programs with negation as failure, but the rule bodies may also contain *queries to* L in their bodies.

Suppose $\Phi = (\mathcal{P}, \mathcal{C})$, is a vocabulary of finite sets $\mathcal{P}$ and $\mathcal{C}$ of predicate and constant symbols, respectively, and a set $\mathcal{X}$ of variables. As usual, elements from $\mathcal{C} \cup \mathcal{X}$ are *terms*, and atoms have the form $p(t_1, \ldots, t_n)$, where $p \in \mathcal{P}$ has arity n and all t_i are terms.

Queries to L occur in so-called dl-atoms. A *dl-query* $Q(\mathbf{t})$ is either

(a) a concept inclusion axiom F or its negation $\neg F$; or
(b) of the forms $C(t)$ or $\neg C(t)$, where C is a concept, and t is a term; or
(c) of the forms $R(t_1, t_2)$ or $\neg R(t_1, t_2)$, where R is a role, and t_1 and t_2 are terms.

A *dl-atom* has then the form

$$DL[S_1 op_1 p_1, \ldots, S_m op_m\ p_m; Q](\mathbf{t}), \qquad m \geq 0, \tag{1}$$

where each S_i is either a concept or a role; $op_i \in \{\uplus, \cup\!\!\!\!-\}$; p_i is a unary (resp.,binary) predicate symbol, if S_i is a concept (a role); and $Q(\mathbf{t})$ is a dl-query. Intuitively, $op_i = \uplus$ (resp. $\cup\!\!\!\!-$) increases S_i ($\neg S_i$) by the extension of p_i.

oarieli@mta.ac.il

A *dl-rule* r is of the form

$$a \leftarrow b_1, \ldots, b_k, not\, b_{k+1}, \ldots, not\, b_m,\ m \geq k \geq 0, \tag{2}$$

where a is an atom (the *head*) and each b_i is either an atom or a dl-atom, and not is negation as failure (default negation). A *dl-program* $KB = (L, P)$ consists of a DL knowledge base L and a finite set of dl-rules P; it is *positive*, if P is positive.

Example 1. Let $KB = (L, P)$ where $L = \{C \sqsubseteq D\}$ and P is the set of rules

$$p(a); \quad p(b); \quad q(c);$$
$$s(X) \leftarrow DL[C \uplus p; D](X),\ not\ DL[C \uplus q, C \uplus p; D](X)\ .$$

Intuitively, we extend in the first dl-atom concept C by predicate p and retrieve then all instances from D in this extended ABox. With the second dl-atom we extend C and $\neg C$ by the extensions of q and p, resp. Thus, the intuitive model for this dl-program would be $\{p(a), p(b), q(c), s(a), s(b)\}$.

Semantics. The *Herbrand base* of P, denoted HB_P, is the set of all atoms $p(c_1 \ldots, c_n)$ where $p \in \mathcal{P}$ occurs in P and all c_i are from $\mathcal{C}$. An *interpretation I relative to P* is any subset of HB_P. Such an I *satisfies* (*models*) a ground (i.e.,variable-free) atom or dl-atom a under L, denoted $I \models_L a$, if the following holds:

- $a \in I$, if $a \in HB_P$;
- $L(I; \lambda) \models Q(\mathbf{c})$, where $\lambda = S_1\, op_1\, p_1, \ldots, S_m\, op_m p_m$, $L(I; \lambda) = L \cup \bigcup_{i=1}^{m} A_i(I)$ and, for $1 \leq i \leq m$,
$$A_i(I) = \begin{cases} \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, & \text{if } op_i = \uplus, \\ \{\neg S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}, & \text{if } op_i = \cup, \end{cases}$$
if a is a ground dl-atom $DL[\lambda; Q](\mathbf{c})$.

I satisfies a ground dl-rule r of form (2) if either (i) $I \not\models_L a$, or (ii) $I \not\models_L b_i$ for some $1 \leq i \leq k$ or (iii) $I \models_L b_j$ for some $k < j < m$. I satisfies a dl-program $KB = (L, P)$, denoted $I \models KB$, iff $I \models_L r$ for every rule $r \in ground(P)$, where $ground(P)$ is the set of all ground instances of rules in P (relative to HB_P).

It is easy to see that every positive KB has some model and, like every Horn logic program, a unique minimal (under inclusion $\subseteq$) model, denoted M_{KB}. This model naturally captures the semantics of KB.

Answer Set Semantics. The *answer sets* of a general dl-program $KB = (L, P)$ are defined by a reduction to positive dl-programs. The *(strong) dl-transform* of P relative to L and an interpretation $I \subseteq HB_P$, denoted sP_L^I, results from $ground(P)$ by deleting (i) every dl-rule r such that $I \models_L a$ for some $a \in B^-(r)$, and (ii) the negative bodies of all remaining dl-rules; note that sP_L^I generalizes the Gelfond-Lifschitz reduct P^I [23]. Let $KB^I = (L, sP_L^I)$. Since KB^I is positive, it has a unique minimal (the least) model, denoted $\mathsf{LM}(KB^I)$. Then $I \subseteq HB_P$ is a *(strong) answer set* of KB, if $I = \mathsf{LM}(KB^I)$. We write $KB \models a$ for a ground atom a if $I \models_L a$ for every answer set of KB.

Well-founded Semantics. Define the operator γ_{KB} on interpretations I of KB by $\gamma_{KB}(I) = \mathsf{LM}(KB^I)$. As γ_{KB} is anti-monotone, $\gamma_{KB}^2(I) = \gamma_{KB}(\gamma_{KB}(I))$ is monotone and has a least fixpoint, which is the set of *well-founded* atoms of KB, denoted $WFS(KB)$ [16]; we denote with $KB \models_{wf} a$ that $a \in WFS(KB)$.

oarieli@mta.ac.il

Example 2. KB from Example 1 has the single answer set $\{p(a), p(b), q(c), s(a), s(b)\}$, which coincides with $WFS(KB)$. If we replace the facts for p in P by the "guessing" rules $p(a) \leftarrow not\ p(b)$; $p(b) \leftarrow not\ p(a)$, the resulting KB has the two answer sets $\{p(a), q(c), s(a)\}$ and $\{p(b), q(c), s(b)\}$, while $q(c)$ is the only well-founded atom.

3 First-Order Rewritability

In this section, we introduce the first-order rewritable case of dl-programs. This case is motivated by the already introduced DL-Lite family, which allows the uniform evaluation of restricted classes of dl-programs on relational database technology.

3.1 First-Order Rewritable dl-Programs

We understand FO rewritability of a dl-program $KB = (L, P)$ in the sense that query answering from KB, i.e., to decide whether $KB \models p(\boldsymbol{c})$ for an atom $p(\boldsymbol{c})$, is expressible by a FO formula $\phi(\boldsymbol{x})$ over the *relational schema* induced by the vocabulary of L, such that $KB \models p(\boldsymbol{c})$ iff $\mathcal{A} \models \phi(\boldsymbol{c})$, where ϕ depends on p, P and L; in a data-centric view, it only depends on the TBox of L, but not on the concrete ABox $\mathcal{A}$. For the DL-Lite family, and in particular for DL-Lite$_R$, the analog property for answering conjunctive queries is also called *FOL-reducibility*.[4] Like for the latter, query answering for FO rewritable dl-programs is feasible on a Relational Data Base Management System (RDBMS) by casting the FO formula into SQL statements.

As recursion is not expressible in FO logic, it must be banned for FO rewritability from KB; this is achieved by acyclicity. Let $\mathcal{P}_P$ be the set of all predicates symbols in P. Then P is acyclic, if there exists a mapping $\mathcal{K}\colon \mathcal{P}_P \to \{0, \ldots, n\}$ such that for every rule $r \in P$ of form (2) and $1 \leq i \leq n$, it holds that $\mathcal{K}(p) > \mathcal{K}(q)$ where p occurs in a and q occurs in b_i (in case of a dl-atom of form (1), q has to occur among the p_j). Note that every acyclic KB has a unique answer set, which coincides with $WFS(KB)$ [16]. Thus, for every ground atom a, $KB \models a$ iff $KB \models_{wf} a$.

An important result regarding FO rewritability of dl-programs under the well-founded semantics was given in [16]. We recall the results in Theorem 1 and 2.

Theorem 1 (FO rewritable dl-programs [16]). *Let $KB = (L, P)$ be an acyclic dl-program, and $p(\boldsymbol{c})$ an atom, such that (1) every dl-query in P is FO-rewritable, and (2) if the operator $\uplus$ occurs in P, then L is defined over a DL that (2a) is* CWA-satisfiable *(i.e., for every DL KB L', the union of L' and all membership assertions that are not entailed by L' is satisfiable), and (2b) allows for FO-rewritable concept and role memberships. Then, deciding $KB \models_{wf} p(\boldsymbol{c})$ is FO-rewritable.*

The proof is based on an induction on mapping $\mathcal{K}$ and the following assumptions:

(a) every dl-atom δ can be expressed as FO formula over the ABox of L;
(b) every predicate of rank 0 is easily expressed as FO formula over the facts of P;
(c) every other predicate p_I can be expressed by the disjunction of the existentially quantified bodies of the rules which share p_I in their heads, and where the NAF atoms are interpreted as classical negation.

[4] Conjunctive queries over dl-programs can be expressed by rules, thus atomic queries suffice.

oarieli@mta.ac.il

Concerning (a), let $\delta = DL[\lambda; Q](\boldsymbol{c})$ be a dl-atom of form (1) such that $\lambda = \lambda^+, \lambda^-$ is the list of m positive ($op_i = \uplus$) and negative ($op_i = \cup\!\!\!-$) extensions of L, and $Q(\boldsymbol{c})$ a dl-query. Each extension $S_i\ op_i\ p_i$ in λ can be expressed in terms of a FO formula $\psi_{S_i}(\boldsymbol{y})$ over L. The dl-query $Q(\boldsymbol{c})$ can be expressed as a FO formula $\alpha(\boldsymbol{x})$ over L. Every input predicate p_j in λ is as a FO formula $\psi_j(\boldsymbol{x})$ over P. Then, the FO formula $\delta(\boldsymbol{x})$ over the ABox $\mathcal{A}$ of L and facts of P

$$\delta(\boldsymbol{x}) = \alpha^{\lambda^+}(\boldsymbol{x}) \vee \bigvee_{j=1}^{m} \exists \boldsymbol{y} \left(\psi_{S_j}^{\lambda^+}(\boldsymbol{y}) \wedge \psi_j(\boldsymbol{y}) \right) \quad , \tag{3}$$

where α^{λ^+} (resp. $\psi_{S_j}^{\lambda^+}$) is obtained from α (resp. ψ_{S_j}) by replacing every $S_i(s)$, such that S_i occurs in λ^+ by $S_i(s) \vee \psi_{i_1}(s) \vee \cdots \vee \psi_{i_{k_i}}(s)$, where $S_{i_1}, \ldots, S_{i_{k_i}}$ are all occurrences of S_j in λ^+. For the meaning of the assumptions, we refer to [16].

Example 3. Consider $KB = (L, P)$ from Example 1. To illustrate all parts of the transformation, we create a variant of KB as $KB' = (L, P')$, where $P' = P \cup \{s(X) \leftarrow q(X)\}$. We can express query $D(X)$ (after the *perfect rewriting*) by the FO formula $\alpha(X) = C(x) \vee D(x)$ over $\mathcal{A}$ ($= \emptyset$). As FO formulas, p and q are $\psi_p = p(x)$ and $\psi_q = q(x)$ over $F = \{p(a), p(b), q(c)\}$. Then the dl-atom $DL[C \uplus p; D](X)$ is translated into $\delta_1(x) = C(x) \vee D(x) \vee p(x)$, while the dl-atom $DL[C \uplus q; C \cup\!\!\!- p; D](X)$ is translated into $\delta_2(x) = C(x) \vee D(x) \vee q(x) \vee \exists y((C(y) \vee q(y)) \wedge p(y))$, both over F. The rules for predicate s are translated into $\exists x.q(x) \vee \exists x.(\delta_1(x) \wedge \neg\delta_2(x))$ over F.

Theorem 2 (FO-rewritable dl-program over the DL-Lite family [16]). *For any vocabulary Φ, acyclic dl-program $KB = (L, P)$, and atom $p(\boldsymbol{c})$, such that 1. L is in a DL of the DL-Lite family, and 2. all dl-queries in P are of the form $C \sqsubseteq D$, $\neg(C \sqsubseteq D)$, $C(t)$ or $R(t, s)$, where C is an atomic concept, D is an (possibly negated) atomic concept, and t, s are terms of L, deciding $KB \models_{wf} p(\boldsymbol{c})$ is first-order rewritable.*

Since the DL-Lite family is CWA-satisfiable [8], operator $\cup\!\!\!-$ is allowed in P and dl-queries of the form $C(t)$ and $R(t, s)$ are immediately FO rewritable. Furthermore, dl-queries of the form $C \sqsubseteq D$ can be reduced to queries as follows: $L' \models C \sqsubseteq D$ iff $L' \cup \{C(e), D'(e), D' \sqsubseteq \neg D, A'(d), A' \sqsubseteq \neg A\} \models A(d)$. Similar, $\neg(C \sqsubseteq D)$ can be reduced to $L' \models \neg(C \sqsubseteq D)$ iff $L' \cup \{C \sqsubseteq D, A'(d), A' \sqsubseteq \neg A\} \models A(d)$, where d and e are fresh individuals, and A, A', and D' are fresh atomic concepts.

3.2 Implementation and Experiments

Based on the ideas above, the experimental system MOR evaluates conjunctive queries over an acyclic dl-program KB using an RDBMS (which we call the database), viz. PostgreSQL 8.4. MOR has three main modules: a Datalog-to-SQL rewriter, a DL-Lite plugin, and an adaption of the DL-Lite$_R$ reasoner Owlgres (see [48]).

- The Datalog-to-SQL rewriter, which is based on well-known techniques, cf. [49], can also handle limited rule recursion (see Subsection 3.3). However, different from DLV^{DB}[49], SQL views are not materialized and recursion is handled differently. In MOR the focus is merely on linear recursion based on a *direct evaluation algorithm* of the transitive closure, and relies on the native implementation of the

oarieli@mta.ac.il

RDBMS. On the other hand in DLV^{DB}, full recursion is implemented based on an optimized *semi-naive algorithm*, where iteratively SQL statements are executed until a fixpoint is reached (in [43] both algorithms are compared).

– The DL-Lite plugin transforms dl-atoms according to the rewriting above. In that, it exploits a modified version of Owlgres to obtain the result of the PerfectRef algorithm [8], i.e., the *perfect rewriting* of a query and the TBox, without execution.

Given $KB = (L, P)$ and a conjunctive query Q, the rewriting puts the facts of P and the ABox of L in the database and rewrites the rules of P into cascading VIEWS. For every predicate p occurring in some rule head, one VIEW$_p$ is created, consisting of the UNION of SELECT−PROJECT−JOIN (SPJ) statements for the bodies of rules with p in the head, where negated atoms $not\ a$ are cast to NOT IN $(\cdot)$ statements. Acyclicity of KB ensures a proper evaluation order of all views; for details, see [46]. The conjunctive query $Q(\boldsymbol{x}) = p_1(\boldsymbol{x}_1), \ldots, p_m(\boldsymbol{x}_m)$, which can be seen as a rule $Q(\boldsymbol{x}) \leftarrow p_1(\boldsymbol{x}_1), \ldots, p_m(\boldsymbol{x}_m)$, is rewritten into a single (SPJ) SQL statement.

With the $\uplus$ and $\uminus$ operator and a *static* ABox in the RDBMS, the rewriting of dl-atoms is more involved. We have to modify the ABox $\mathcal{A}$ temporally to $\mathcal{A}'$ prior to evaluating the dl-query over $\mathcal{A}'$. In addition, the internal DB schema of Owlgres must be respected. Again, acyclicity of KB ensures an evaluation order $\mathcal{K}$ for the temporary modified ABoxes, which is realized for each $DL[\lambda; Q](c)$ as follows:

(a) create new VIEWS representing ABoxes $\mathcal{A}'_{\mathcal{K}}$, building the union of $\mathcal{A}'_{\mathcal{K}-1}$ and all $S_i\ op_i\ p_i$ of λ (where p_i is an existing view);
(b) Modify the perfect rewriting of Q to use $\mathcal{A}'_{\mathcal{K}}$ instead of $\mathcal{A}$.

Clearly, after the evaluation of the dl-program, $\mathcal{A}$ has to be reverted to its original state. MOR allows to access other plugins than the DL-Lite plugin, which may support dl-atoms for querying other DLs, or even other kinds of knowledge sources (e.g., a spatial database). The basic requirement is that the plugin has to return an SQL rewriting.

For experimentation, we considered three different categories of benchmark instances:

(1) randomly generated sets of facts (R_n); (2) a simplified version of DBpedia (D_n); and (3) the well-known Lehigh University Benchmark (LUBM) [26] (U_n). As LUBM is not fully in DL-Lite$_R$, we altered roughly 10% of the TBox axioms like transitive roles to normal roles and equality axioms of the form $B \equiv C_1 \sqcap C_2$ to $B \sqsubseteq C_1$ and $B \sqsubseteq C_2$. The index n in our instances denote the number of facts in R_n and the ABox assertions in D_n and U_n; we report the outcome for runs with $n \in \{10\mathrm{k}, 100\mathrm{k}, 250\mathrm{k}, 500\mathrm{k}, 1\mathrm{M}\}$. The test data of R_n was randomly generated allowing a high selectivity among the join attributes. For D_n, different sets of books, periodicals, and publications were extracted from DBpedia, including a single role. The test data for U_n was generated by the LUBM instance generator; e.g., university $U_{100\mathrm{k}}$ has about 12k individuals.

As a baseline system, we compared MOR to DLV [38], DLV^{DB}[49], and dlvhex [18]. DLV^{DB} is a tight coupling of DLV with a relational DBMS, in which SQL queries over an (external) database can be evaluated. In case of dlvhex, we used its standard DL plug-in (interfacing RacerPro 1.9.2 [27]), which we refer to as dlvhex[DL]. The benchmark runs FO_1–FO_3 are summarized in Table 1. Here, FO_1 has no ontology access; it serves to assess the rule rewriting. In FO_2, the dl-program extended the ABox by constantly

oarieli@mta.ac.il

Table 1. Benchmark Overview

Name	Description	Systems	Data	Reference
FO_1	Tree of binary joins (with negation)	DLV; DLV^{DB}	Random	[46, Ex. 5.2.1]
FO_2	Select a range of the KB upon extension with books from an external source	DLV^{DB}; dlvhex[DL]	DBpedia	[46, Ex 5.3.3]
FO_3	Seek students taking courses of faculty advisors who are not full professors	dlvhex[DL]	LUBM	[46, Ex 5.4.2]
FO_4	Transitive closure of the organization hierarchy fed to the DL KB	dlvhex[DL]	LUBM	[46, Ex 5.4.3]

Table 2. Benchmark Results for FO-Rewritable DL-Programs (Runtime in secs)

(a) Benchmark FO_1

Instance	MOR	DLV^{DB}	DLV
R_{10k}	<1	<1	1
R_{100k}	1	1	105
R_{250k}	3	4	977
R_{500k}	5	9	2,795
R_{1M}	11	19	11,446

(b) Benchmark FO_2

Instance	MOR	DLV^{DB}	dlvhex[DL]
D_{10k}	1	<1	7
D_{100k}	4	6	—
D_{250k}	9	25	—
D_{500k}	18	50	—
D_{1M}	42	145	—

(c) Benchmark FO_3

Instance	MOR	dlvhex[DL]
U_{10k}	1	36
U_{100k}	4	117
U_{250k}	11	—
U_{500k}	20	—
U_{1M}	44	—

(d) Benchmark FO_4

Instance	MOR	dlvhex[DL]
U_{10k}	1	35
U_{100k}	1	108
U_{250k}	2	—
U_{500k}	4	—
U_{1M}	11	—

60 individuals, imported from a randomly generated relation. The program in the runs of FO_3 resembles Query 9 of [26] with a triangular pattern of low selectivity roles plus an additional NAF atom.

We conducted our experiments on an openSUSE 11.1 (x86_64) server having a Intel Xeon CPU E5450 3.00GHz and 15.7 GB of RAM. The rewriting was evaluated on a PostgreSQL 8.4 database, with increased `shared_buffers` and `work_mem` parameters to utilize the available RAM. For each benchmark, the average of five runs was calculated, having a timeout of 6 hours, and a memout of 14.7 GB for each run.

The results are shown in Table 2; the entries with "—" (*out of memory*) for dlvhex are due to RacerPro's usage of the AllegroGraph library, which limited the instance size. (The optimized RacerPro 2.0 release is not available with dlvhex at present.) The results let observe a linear runtime behavior of MOR for all benchmarks except FO_4. The use of indexes in the database might lead to performance gains. Owlgres creates default indexes during KB loading, but none exploiting the particular dl-program structure. In

oarieli@mta.ac.il

case of FO_1 (Table 2a), MOR and DLV^{DB} scale similarly, but the use of views is faster than materialization as applied in the standard version of DLV^{DB}. On the other hand, the rule rewriting of DLV^{DB} appears to be effective. In case of FO_2 (Table 2b), the temporary update of the DL knowledge base did not hit much on MOR's performance and also here quasi-linear runtime was achieved; DLV^{DB} also scales well but at a more increasing pace. Further details are available on the benchmark webpage.[3]

3.3 Limited Recursion

FO rewritability is mainly motivated by its link to RDBMS and SQL, which allows for exploiting efficient database engines. The SQL:1999 standard, however, also foresees a limited form of linear recursion in queries, such that, e.g., the transitive closure of a base relation is definable. The respective classic rule-based definition for base relation a

$$p(X,Y) \leftarrow a(X,Y). \qquad\qquad p(X,Y) \leftarrow a(X,Z),\ p(Z,Y).$$

is written in SQL as

```
WITH RECURSIVE p AS
(SELECT * FROM a  UNION  SELECT a.1 p.2 FROM a, p WHERE a.2 = p.1)
SELECT * FROM p
```

To exploit SQL:1999 linear recursion features for dl-programs, we impose in addition to linearity the following syntactic restriction: (1) no predicate depends negatively on itself (i.e., negation is stratified), and (2) no predicate cycles through some dl-atom; the latter still allows recursive predicates as input of dl-atoms. Informally, this supports linear stratified negation with non-recursive DL KB access.

Related to the experiments above, we performed benchmark FO_4 based on the LUBM ontology U_n and a linear recursive dl-program. As DL-Lite$_R$ misses transitive roles, we calculate the transitive closure of the sparse, tree-like structure of the $subOrganization$ role (thus the organizational hierarchy of the LUBM university) in the rules part and update the DL KB. We observed in the results (in Table 2d) an indication for quadratic runtime. We also point to another experiment (see [46, Ex. 5.2.3]), where randomly generated, cyclic data led to irregular runtime; in one case, runs with $U_{50\mathrm{k}}$ were faster than with $U_{25\mathrm{k}}$ due to depth and cycles. We further encountered a limitation in Postgres' recursive query implementation on cyclic data. As it iterates joins without cycle detection,[5] queries even may not terminate. To avoid this, we used Postgres's LIMIT parameter to safely bound the iterations.

4 Datalog$^{\neg}$ Rewritability

In this section, we consider uniform evaluation of dl-programs by rewriting to Datalog$^{\neg}$, which basically are dl-programs as in Section 2.2 but without dl-atoms. Positive such programs are known as *plain* Datalog and have a canonical semantics, while under negation different semantics (answer sets, well-founded atoms) are widely used.

[5] http://archives.postgresql.org/pgsql-hackers/2008-02/msg00642.php

oarieli@mta.ac.il

Datalog$^\neg$ is more expressive than FO logic, and compared with FO rewriting, we do not pose any restriction in the rule part, thus we can naturally use recursive rules. The ontologies in dl-programs can also be "inlined" into some Datalog$^\neg$ programs in a modular way.

4.1 DL Datalog Rewritability and Inline Evaluation

For Datalog$^\neg$ rewritability of dl-programs, we need a suitable notion of rewritability of a DL knowledge base. To this end, [28] defined a description logic $\mathcal{DL}$ to be Datalog-*rewritable*, if there exists a transformation $\Phi_{\mathcal{DL}}$ from $\mathcal{DL}$ KBs to Datalog programs such that, for every $\mathcal{DL}$ KB L,

(i) $L \models Q(\mathbf{o})$ iff $\Phi_{\mathcal{DL}}(L) \models Q(\mathbf{o})$ for any concept or role name Q from L, and individuals $\mathbf{o}$ from L;
(ii) $\Phi_{\mathcal{DL}}$ is *modular*, i.e., for $L = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox, $\Phi_{\mathcal{DL}}(L) = \Phi_{\mathcal{DL}}(\mathcal{T}) \cup \mathcal{A}$;

DL-programs $KB = (L, P)$ over Datalog-rewritable DLs can be readily transformed to Datalog$^\neg$ programs. Let $\Lambda_P = \{\lambda \mid DL[\lambda; Q]$ occurs in $P\}$ be the set of all input lists of dl-atoms appearing in P. The transformation $\Psi(KB)$ of a dl-program KB is then defined as $\Phi_{\mathcal{DL}}(L_{\Lambda_P}) \cup P^{ord} \cup \rho(\Lambda_P) \cup T_P$, where

- $L_{\Lambda_P} = \bigcup_{\lambda \in \Lambda_P} L_\lambda$, where L_λ is L with all concept and role names subscripted with λ. Intuitively, each input signature of a dl-atom in P will influence L differently. As we want to cater for these influences in one program, we have to differentiate between the KBs with different inputs;
- $\rho(\Lambda_P)$ is a Datalog program that contains for each $\lambda = S_1 \uplus p_1, \ldots, S_m \uplus p_m \in \Lambda_P$ the rules $S_{i\lambda}(\mathbf{X_i}) \leftarrow p_i(\mathbf{X_i})$, $1 \leq i \leq m$, where the arity of $\mathbf{X_i}$ matches the one of S_i. Intuitively, we add the extension of p_i to the appropriate concept or role;
- P^{ord} is P with each dl-atom $DL[\lambda; Q](\mathbf{t})$ replaced by a new atom $Q_\lambda(\mathbf{t})$;
- T_P consists of Datalog facts $\top(a)$ and $\top^2(a, b)$ for all a, b in the Herbrand domain of P to ensure their introduction in L.

Example 4. Consider $KB' = (L, P')$, which replaces the last rule of P in Example 1 by $s(X) \leftarrow DL[C \uplus p; D](X), not\ DL[C \uplus q; D](X)$. Here, L is rather simple. We can transform L_{Λ_P} to $\Phi(L_{\Lambda_P}) = \{D_{\lambda_1}(X) \leftarrow C_{\lambda_1}(X);\ D_{\lambda_2}(X) \leftarrow C_{\lambda_2}(X)\}$. Then $\Lambda_P = \{\lambda_1 = C \uplus p, \lambda_2 = C \uplus q\}$ and $\rho(\Lambda_P) = \{C_{\lambda_1}(X) \leftarrow p(X);\ C_{\lambda_2}(X) \leftarrow q(X)\}$. Program P^{ord} consists of all facts of P' and the rule $s(X) \leftarrow D_{\lambda_1}(X), not\ D_{\lambda_2}(X)$. Finally, we add $T_P = \{\top(o) \mid o \in \{a, b, c\}\} \cup \{\top^2(o_1, o_2) \mid \{o_1, o_2\} \subset \{a, b, c\}\}$.

The following result allows us to reduce reasoning from dl-programs to Datalog$^\neg$ under well-founded semantics.

Theorem 3 ([28]). *Let $KB = (L, P)$ be a dl-program over a* Datalog-*rewritable DL and $a \in HB_P$. Then, $KB \models_{wf} a$ iff $\Psi(KB) \models_{wf} a$.*

A similar result holds for answer set semantics.

Theorem 4. *Let KB be a dl-program over a* Datalog-*rewritable DL. Then the answer sets of KB correspond 1-1 to the answer sets of $\Psi(KB)$, such that (i) every answer*

oarieli@mta.ac.il

set of KB is expendable to an answer set of $\Psi(KB)$; and (ii) for every answer set J of $\Psi(KB)$, its restriction $I = J\,|_{HB_P}$ to HB_P is an answer set of KB.

The reduction from dl-programs to Datalog$^\neg$ above only considers the operation $\uplus$ and positive dl-queries. Negative dl-queries can be reduced to inconsistency checking (see Sec. 4.3 for a discussion). Next we consider concrete datalog rewritable DLs.

4.2 Inline Evaluation over $\mathcal{LDL}^+$ Ontologies

The description logic $\mathcal{LDL}^+$ [28,55] is designed as a lightweight ontology language which is expressive enough to capture many real life ontologies. It imposes syntactic restrictions on axioms $\alpha \sqsubseteq \beta$, distinguishing between the "body" α and the "head" β, shown in Table 3a.

An $\mathcal{LDL}^+$ ontology is a pair $L = \langle \mathcal{T}, \mathcal{A} \rangle$ of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, where

- $\mathcal{T}$ is a set of *terminological axioms* $B \sqsubseteq H$, where B is a b-concept and H is an h-concept, and *role axioms* $S \sqsubseteq T$, where S is a b-role and T is an h-role, and
- $\mathcal{A}$ is a set of assertions of the form $C(o)$ and $S(o_1, o_2)$ where C is an h-concept and S is an h-role.

Essentially, $\mathcal{LDL}^+$ extends OWL 2 RL [40] with singleton nominals, role conjunctions, and transitive closure. Designing a transformation $\Phi_{\mathcal{LDL}^+}$ of $\mathcal{LDL}^+$ into Datalog is straightforward; each TBox (ABox) axiom is transformed into an equivalent Datalog rule (fact), similarly as in [25]. For example, if $Person \sqcap \exists headOf.Dept \sqsubseteq Chair \in L$, then we add a rule $Chair(X) \leftarrow Person(X), headOf(X, Y), Dept(Y)$ to the Datalog program $\Phi_{\mathcal{LDL}^+}(L)$. For details of the rewriting, please see [28].

4.3 Inline Evaluation over EL Ontologies

We consider the DL $\mathcal{SROEL}(\sqcap, \times)$ [37], whose syntax is given in Table 3b; it is a superset of OWL 2 EL [40] disregarding datatypes, and adds concept production, which can be seen as a generalization of domain and role restriction.

Krötzsch [37] shows a Datalog encoding for $\mathcal{SROEL}(\sqcap, \times)$ describing a proof system. Every TBox axiom, ABox axiom, concept name, role name, and individual is transformed to a *fact* by an input translation I_{inst}. A fixed set P_{inst} contains the derivation rules, which are independent of the concrete $\mathcal{SROEL}(\sqcap, \times)$ ontology (see [37]).

In the following, for simplicity, when we say $\mathcal{EL}$, we always mean $\mathcal{SROEL}(\sqcap, \times)$. For an $\mathcal{EL}$ ontology L, define a Datalog transformation by

$$\Phi_{\mathcal{EL}}(L) = P_{inst} \cup \{I_{inst}(\alpha) \mid \alpha \in L\} \cup \{I_{inst}(s) \mid s \in N_I \cup N_C \cup N_R\} \ .$$

Theorem 5 ([37]). *Given an $\mathcal{EL}$ ontology L, the transformation $\Phi_{\mathcal{EL}}$ is sound and complete w.r.t. instance checking, i.e., (i) $L \models C(a)$ iff $\Phi_{\mathcal{EL}}(L) \models isa(a, C)$, and (ii) $L \models R(a, b)$ iff $\Phi_{\mathcal{EL}}(L) \models triple(a, R, b)$.*

Example 5. Let $L_1 = \{A(a),\ A \sqsubseteq \exists R.B,\ B \sqsubseteq C,\ \exists R.C \sqsubseteq D\}$, and suppose we want to decide $L_1 \models D(a)$. The axioms of L_1 and the signatures N_I, N_C, and N_R are transformed to facts in $\Phi_{\mathcal{EL}}(L_1)$:

oarieli@mta.ac.il

Table 3. Syntax of $\mathcal{LDL}^+$ and $\mathcal{SROEL}(\sqcap, \times)$

(a) head (h-) and body (b-) restrictions on roles and concepts in $\mathcal{LDL}^+$ axioms

- *h-roles* (h for *head*) S, T are (i) *role names* R, (ii) *role inverses* S^-, (iii) *role conjunctions* $S \sqcap T$, and (iv) *role top* $\top^2$;
- *b-roles* (b for *body*) S, T are the same as h-roles, plus (v) *role disjunctions* $S \sqcup T$, (vi) *role sequences* $S \circ T$, (vii) *transitive closures* S^+, (viii) and *role nominals* $\{(o_1, o_2)\}$, where o_1, o_2 are individuals.
- *basic concepts* C, D are concept names A, $\top$, and conjunctions $C \sqcap D$;
- *h-concepts* are (i) *basic concepts* B, and (ii) *value restrictions* $\forall S.B$ where S is a *b-role*;
- *b-concepts* C, D are (i) *basic concepts* B, (ii) *disjunctions* $C \sqcup D$, (iii) *exists restrictions* $\exists S.C$, (iv) *atleast restrictions* $\geq nS.C$, and (v) *nominals* $\{o\}$, where S is a b-role, and o is an individual.

(b) Syntax of $\mathcal{SROEL}(\sqcap, \times)$

- Concept constructors are (i) top $\top$, (ii) bottom $\bot$, (iii) conjunction $C \sqcap D$, (iv) existential restriction $\exists R.C$, (v) nominal $\{a\}$;
- Axioms are (i) concept assertion $C(a)$, (ii) role assertion $R(a, b)$, (iii) concept inclusion (GCI) $C \sqsubseteq D$, (iv) role inclusion $R \sqsubseteq T$, (v) generalized role inclusion $R \circ S \sqsubseteq T$, (vi) role conjunction $S_1 \sqcap S_2 \sqsubseteq T$, (vii) concept production $C \times D \sqsubseteq T$, $R \sqsubseteq C \times D$.

$$\left\{ \begin{array}{l} isa(a, A);\ supEx(A, R, B, e^{A \sqsubseteq \exists R.B});\ subClass(B, C);\ subEx(R, C, D); \\ nom(a);\ cls(A);\ cls(B);\ cls(C);\ cls(D);\ rol(R) \end{array} \right\} .$$

Then, P^{inst} is added to $\Phi_{\mathcal{EL}}(L_1)$; in particular, also the rules

$$\begin{aligned} isa(X, X) &\leftarrow nom(X) \\ isa(X, Z) &\leftarrow subClass(Y, Z), isa(X, Y) \\ isa(X_1, Z) &\leftarrow subEx(V, Y, Z), triple(X_1, V, X_2), isa(X_2, Y) \\ triple(X_1, V, X_2) &\leftarrow supEx(Y, V, Z, X_2), isa(X_1, Y) \\ isa(X_2, Z) &\leftarrow supEx(Y, V, Z, X_2), isa(X_1, Y) \end{aligned}$$

From these rules and the above facts, $isa(a, D)$ is derivable, and thus $\Phi_{\mathcal{EL}}(L_1) \models D(a)$.

Note that strictly, $\Phi_{\mathcal{EL}}(L)$ is not a datalog rewriting as defined above. The mismatch is that ABox assertions (e.g., $C(a)$) are transformed into reified versions (e.g., $isa(a, C)$); this is easily fixed by using reification rules

$$\begin{aligned} P^{re} = \{&C(X) \leftarrow isa(X, C); \quad isa(X, C) \leftarrow C(X) \mid C \in N_C\} \cup \\ &\{R(X, Y) \leftarrow triple(X, r, Y); \quad triple(X, R, Y) \leftarrow r(X, Y) \mid R \in N_R\} \ . \end{aligned}$$

Then $\Phi'_{\mathcal{EL}}(L) = (\Phi_{\mathcal{EL}}(L) \setminus \{I_{inst}(\alpha) \mid \alpha \in \mathcal{A}\}) \cup P^{re} \cup \mathcal{A}$ is a proper Datalog rewriting. However, in the following, for convenience, we will use $\Phi_{\mathcal{EL}}$, instead of $\Phi'_{\mathcal{EL}}$.

Negative dl-Queries. Trivially, $L \models \neg C(a)$ is equivalent to unsatisfiability of $L \cup \{C(a)\}$. To answer a negative query $\neg C(X)$, we need to bind X to every possible

oarieli@mta.ac.il

individual, and reduce it to unsatisfiability checking. This one by one checking can be elegantly achieved via datalog encoding. The idea is to extend $isa/2$ with two more arguments, representing the individual and the concept name, to $isa_n/4$; in P_{inst}, each isa is uniformly replaced with isa_n, and each $triple$ uniformly with $triple_n$, yielding $P^{\neg}_{inst}$. This set includes e.g. the following rules, which propagate subclass and conjunctive subclass membership:

$$isa_n(X, Z, C, J) \leftarrow subClass(Y, Z), isa_n(X, Y, C, J)$$
$$isa_n(X, Z, C, J) \leftarrow subConj(Y_1, Y_2, Z), isa_n(X, Y_1, C, J), isa_n(X, Y_2, C, J)$$

The individual unsatisfiability checks are then accomplished with rules $P^{\neg}$, which for each check add $C(a)$, expand all $isa(X, Y)$ atoms with a and C, and make an atom $isnota(a, C)$ true iff the test is successful:

$$\begin{aligned} isa_n(X, Y, Y, X) &\leftarrow nom(X), cls(Y) \\ isa_n(X_1, Y_1, Y_2, X_2) &\leftarrow isa(X_1, Y_1), cls(Y_2), nom(X_2) \\ isnota(X, Y) &\leftarrow isa_n(N, Z, Y, X), nom(N), bot(Z), nom(X) \end{aligned}$$

Let the extended $\mathcal{EL}$ reduction for negative query of an $\mathcal{EL}$ ontology L be defined as $\Phi^{\neg}_{\mathcal{EL}}(L) = P^{\neg}_{inst} \cup \{I_{inst}(\alpha) \mid \alpha \in L\} \cup \{I_{inst}(s) \mid s \in N_I \cup N_C \cup N_R\} \cup P^{\neg}$.

Proposition 1. *For every $\mathcal{EL}$ ontology L, $L \models \neg C(a)$ iff $\Phi^{\neg}_{\mathcal{EL}}(L) \models isnota(a, C)$.*

An application of dl-programs where negative dl-queries are important is terminological default logic over DLs [5,11]. Here, Reiter-style default rules are applied to named individuals; we show the classic birds&penguins example for illustration.

Example 6. The knowledge base $\Delta = \langle L, D\rangle$ consists of a DL KB $L = \{Flier \sqcap NonFlier \sqsubseteq \bot, Penguin \sqsubseteq Bird,\ Penguin \sqsubseteq NonFlier,\ Bird(tweety)\}$ (which is in $\mathcal{EL}$), and a (singleton) set $D = \{Bird(X) : Flier(X)/Flier(X)\}$ of default rules (informally, birds fly by default). The semantics of the KB Δ is captured by the following dl-program $KB = (L, \Pi(D))$ under answer set semantics (i.e., default extensions correspond to answer sets), where $\Pi(D)$ is

$$\begin{aligned} in_Flier(X) &\leftarrow not\ out_Flier(X) \\ out_Flier(X) &\leftarrow not\ in_Flier(X) \\ Flier^{+}(X) &\leftarrow DL[\lambda; Bird](X), not\ \ DL[\lambda'; \neg Flier](X) \\ fail &\leftarrow DL[\lambda'; Flier](X), out_Flier(X), not\ fail \\ fail &\leftarrow DL[\lambda; Flier](X), in_Flier(X), not\ fail \\ fail &\leftarrow DL[\lambda; Flier](X), out_Flier(X), not\ fail \end{aligned}$$

where $\lambda = \{Flier \uplus in_Flier\}$ and $\lambda' = \{Flier \uplus Flier^{+}\}$. To rewrite KB to Datalog$^{\neg}$, we use for the dl-atom $DL[\lambda'; \neg Flier](X)$ the rewriting $\Phi^{\neg}_{\mathcal{EL}}(L_{\lambda'}) \cup \rho(\lambda') \models isnota_{\lambda'}(X, Flier)$, and for the other dl-atoms the rewriting $\Phi_{\mathcal{EL}}(L_{\gamma}) \cup \rho(\gamma) \models isa_{\gamma}(X, Flier)$, $\gamma \in \{\lambda, \lambda'\}$. The Datalog$^{\neg}$ program has a single answer set, which contains $Flier(tweety)$, as expected.

4.4 Implementation and Experiments

We implemented the datalog rewriting of dl-programs in the DReW reasoner. Initial results over $\mathcal{LDL}^{+}$ ontologies were given in [56], which compares DReW to dlvhex[DL].

oarieli@mta.ac.il

Table 4. Benchmark Results for Datalog Rewriting (Runtime in secs)

(a) $\mathcal{EL}$ Instance Retrieval with Galen (HT=HermiT; Pellet always timed out after one hour)

Ontology	Query	DReW [DLV]	DReW [clingo]	HT	Ontology	Query	DReW [DLV]	DReW [clingo]	HT
G_1	q_1	2.0	1.3	8.1	G_3	q_1	2.0	1.3	9.5
	q_2	2.0	1.3	8.2		q_2	2.1	1.4	9.5
	q_3	2.0	1.3	8.		q_3	2.0	1.4	9.7
	q_4	2.0	1.4	8.1		q_4	2.1	1.4	9.5
G_2	q_1	2.0	1.3	8.9	G_4	q_1	2.1	1.4	10.3
	q_2	2.0	1.4	8.9		q_2	2.1	1.4	10.2
	q_3	2.1	1.4	8.7		q_3	2.1	1.4	10.2
	q_4	2.0	1.3	9.0		q_4	2.1	1.4	10.2

(b) Default reasoning with Policy Benchmark

KB	Typing	DReW [DLV]	DReW [clingo]
Δ_1	5	1.1	0.8
	50	2.4	1.3
	100	6.0	3.0
Δ_5	5	6.6	4.4
	50	8.3	5.0
	100	12.2	7.4
Δ_{10}	5	13.9	9.4
	50	15.7	10.1
	100	20.5	13.3
Δ_{25}	5	35.8	26.0
	50	40.0	26.4
	100	43.7	32.7

The results clearly show that DReW outperforms the latter. For the present paper, we extended the rewriter to the $\mathcal{EL}$ encoding given above. Thus, DReW can now be used as a DL-reasoner and a dl-program-reasoner for $\mathcal{LDL}^+$ and $\mathcal{EL}$ ontologies.

The experiments have been run on an Ubuntu Linux 11.10 system on an AMD Opteron Magny-Cours 6176 SE 2.3GHz system with 24 cores and 128GB RAM. Further details are given on the benchmark webpage.[3]

Instance retrieval With Large $\mathcal{EL}$ TBoxes. The experiment shows the efficiency of the $\mathcal{EL}$ datalog rewriting for a large TBox. As test ontology we used an $\mathcal{EL}$ variant of Galen,[6] a large biomedical ontology.[7] In the test, we created four ontology instances G_1 to G_4 that have a fixed Galen TBox and increasing ABoxes with $10i$ assertions, each using roughly ten concepts and roles. We performed four instance retrieval queries over G_i: (i) $q_1(X) = Substance(X)$, (ii) $q_2(X) = Animal(X)$, (iii) $q_3(X) = MaleAdult(X)$, and (iv) $q_4(X) = Human(X)$.

We were using two Datalog engines for computing the models of the DReW encodings: clingo 3.0.3 [22] and DLV 2010-10-14 [38]. We compared DReW with the DL reasoners HermiT 1.3.5 [42] and Pellet 2.3.0 [47]. We differentiate two DReW benchmark run settings: (i) DReW[clingo] and (ii) DReW[DLV] (in brackets the employed model builder is given). The results are shown in Table 4a (the results for Pellet are omitted as it timed out in all tests).

DReW is superior to HermiT and Pellet; even with small ABoxes, the general purpose DL-reasoner Pellet could not answer the queries within one hour. Note that other reasoners like CB [33] and ELK [34] classify EL-Galen very fast, but they can not be used for instance retrieval out of the box.

[6] `http://condor-reasoner.googlecode.com/files/EL-GALEN.owl`
[7] `http://www.opengalen.org/`

oarieli@mta.ac.il

$$\mathcal{T} = \left\{ \begin{array}{l} Staff \sqsubseteq User, \quad Blacklisted \sqsubseteq Staff, \quad Deny \sqcap Grant \sqsubseteq \bot, \\ UserRequest \equiv \exists hasAction.Action \sqcap \exists hasSubject.User \sqcap \exists hasTarget.Project, \\ StaffRequest \equiv \exists hasAction.Action \sqcap \exists hasSubject.Staff \sqcap \exists hasTarget.Project, \\ BlacklistedStaffRequest \equiv StaffRequest \sqcap \exists hasSubject.Blacklisted \end{array} \right\}$$

$$D = \left\{ \begin{array}{l} UserRequest(X) : Deny(X)/Deny(X), \\ StaffRequest(X) : \neg BlacklistedStaffRequest(X)/Grant(X), \\ BlacklistedStaffRequest(X) : \top/Deny(X) \end{array} \right\}$$

Fig. 1. Access policy control $\Delta = (L, D)$, $L = (\mathcal{T}, \mathcal{A})$, in terminological default logic

Default Reasoning with $\mathcal{EL}$ Policy Ontology. This experiment was conducted on terminological default KBs with an $\mathcal{EL}$ ontology. As shown in Example 6, default reasoning can be encoded into dl-programs with recursive and unstratified rules, which are not easy to handle. For this example, dlvhex[DL] scales exponentially with the number of birds, while DReW, on top of DLV as well as clingo, runs much faster.

As a more interesting benchmark, we consider here an access control policy, borrowed from [6] and couched into a terminological default KB $\Delta = \langle L, D \rangle$, where the the TBox of L and the defaults D are shown in Figure 1; informally, D expresses that users normally are denied access to files, staff is normally granted access to files, while to blacklisted staff any access is denied.

In the test, we created ontology instances L_i, $i \in \{1, 5, 10, 25\}$, that have a fixed TBox and increasing Aboxes with $i{*}1000$ instances of user requests.

The query imposed was then whether a set of particular individuals, designated by concepts Q_k, $k \in \{5, 50, 100\}$, are granted access (under answer set semantics); the application of defaults, using Q_k as a typing concepts, is thus restricted to the k queried individuals. As we see in Table 4b, DReW scales sublinearly in this experiment, on top of both DLV and clingo.

5 Modular ASP

In the transformations of dl-programs to Datalog$^\neg$ above, dl-atoms are encoded by subprograms emulating their evaluation. We can make the modular structure of the programs explicitly visible, if we use an extension of Datalog$^\neg$ which caters for structured programming, e.g., [32]. In modular nonmonotonic logic programs [9], program modules can be defined akin to modules or procedures in common programming paradigms, such that on input of suitable predicates to a module, the valuation of "output" predicates in an answer set of a module is obtained based on a call-by-value mechanism.

For example, a module $kernel[e]$ may compute, in each answer set, a kernel of a graph stored by its edges in a binary predicate e, in a predicate k; here e is a formal parameter, which in a module call must be replaced by a predicate name. E.g., $kernel[my_e].k(a)$ asks if the node a is in the (nondeterministically) computed kernel k of the concrete graph my_e.

A *modular logic program* (MLP) is a collection $\mathbf{P} = (P_1, \ldots, P_n)$ of modules, where each P_i has a list $\boldsymbol{q}_i = q_{i,1}, \ldots, q_{i,n_i}$ of $n_i \geq 0$ formal input parameters; one module (say P_1) is the main module and has $n_0 = 0$. The "implementations" of the

oarieli@mta.ac.il

Table 5. Benchmark dl-programs DReW vs. TD-MLP (Runtime in secs)

(a) Ontology U_1

Program	DReW [clingo]	DReW [DLV]	TD-MLP [clingo]	TD-MLP [DLV]
P_0	0.31	0.45	1.98	2.88
P_1	0.32	0.44	1.69	2.47
P_2	0.32	0.44	2.63	3.82
P_3	0.31	0.43	1.66	2.42
P_4	0.32	0.45	2.45	3.63
P_5	0.61	0.86	1.66	2.46
P_6	1.79	2.76	5.65	8.41
P_7	2.70	4.30	4.87	7.84
P_8	2.76	4.26	9.70	14.12
P_9	2.73	4.31	8.04	11.60

(b) Ontology U_{15}

Program	DReW [clingo]	DReW [DLV]	TD-MLP [clingo]	TD-MLP [DLV]
P_0	6.49	10.27	30.43	42.53
P_1	4.00	6.27	21.22	30.12
P_2	3.95	6.08	32.65	45.24
P_3	3.98	6.13	20.94	30.33
P_4	4.15	6.43	28.19	39.93
P_5	7.97	12.66	21.54	30.87
P_6	23.52	40.56	72.86	103.76
P_7	36.33	64.05	115.03	162.21
P_8	36.58	61.71	128.01	181.41
P_9	35.26	62.19	108.38	145.04

P_i consist of logic program rules of form (2), where the b_i may be atoms or *call atoms* of the form $P_j[\boldsymbol{p}].a'$, where $\boldsymbol{p}$ is a list of predicate names matching $\boldsymbol{q}$ and a' an atom. Answer sets of a MLP are defined in [9] in a natural way, generalizing the answer set semantics of ordinary programs.

We will briefly discuss how the Datalog$^{\neg}$ transformation above can be expressed as MLPs, and examine possible tradeoffs between code repetition and performance, based on a prototype implementation for an MLP fragment.

Inline evaluation of Datalog**-rewritable dl-programs.** To exemplify the rewriting, consider Example 4 above. We can encode $KB = (L, P')$ as an MLP $\mathbf{P} = (P_1[], P_{DL}[C])$, where P_1 is the main module that consists of the rules

$$p(a) \leftarrow \quad p(b) \leftarrow \quad q(c) \leftarrow$$
$$s(X) \leftarrow P_{DL}[p].D(X), not\ P_{DL}[q].D(X) \qquad (r_s)$$

and $P_{DL}[C]$ is a library module consisting simply of the rules $T_P \cup \{D(X) \leftarrow C(X)\}$.

On the surface, the semantics for MLPs creates for the two input lists p and q of our module atoms of P_1 (among others) two instantiations of $P_{DL}[C]$ on-the-fly; the first instantiation for module atom $P_{DL}[p].D(X)$ contains the rules of $P_{DL}[C]$ and the input facts $F_1 = \{C(a), C(b)\}$, and the other instantiation for module atom $P_{DL}[q].D(X)$ has the rules of $P_{DL}[C]$ and input fact $F_2 = \{C(c)\}$. The input F_1 stems from adding the extension of p as C's to $P_{DL}[C]$, while F_2 has been created from the extension of q.

Now the module atom $P_{DL}[p].D(X)$ is retrieving all instances of concept D from $L \cup F_1$ in the instantiation with input F_1. Similarly, $P_{DL}[q].D(X)$ retrieves all instances of D from $L \cup F_2$ in the instantiation with input F_2. The former is true for $P_{DL}[p].D(a)$ and $P_{DL}[p].D(b)$, as we add both $C(a)$ and $C(b)$ to the module, thus the rule $D(X) \leftarrow C(X)$ fires for $X \in \{a, b\}$, simulating the concept inclusion $C \sqsubseteq D$ of L. The ground rules with module atom $P_{DL}[q].D(a)$ and $P_{DL}[q].D(b)$ are false on the other hand, as F_2 only adds $C(c)$ and $D(a)$ and $D(b)$ cannot be derived. Hence, both $not\ P_{DL}[q].D(a)$ and $not\ P_{DL}[q].D(a)$ are true and our rule r_s derives $s(a)$ and $s(b)$ in module $P_1[]$.

oarieli@mta.ac.il

Implementation and experimental results. The algorithm for solving input-call stratified MLPs [10] has been implemented in the TD-MLP solver [54], which is based on the dlvhex system. Using this solver, we could perform initial experiments with the modular datalog encoding sketched in this section. The hardware specification for the experiments are the same as in Section 4.4.

We were testing with dl-programs $KB = (U_i, P_j)$, where U_i ($i \in \{1, 15\}$) are $\mathcal{EL}$ versions of the LUBM ontology [26] and program P_j ($j \in \{0, \ldots, 9\}$) encode variants of the LUBM queries; all dl-programs where acyclic. We denote with U_i the LUBM ontology instance that incorporates i universities in the ABox. The original LUBM is not fully in $\mathcal{EL}$ (inverse roles and datatypes are not part of $\mathcal{EL}$): there are 2 violating axioms in the TBox, and 2857 (33154) ABox axioms with datatype are violated in U_1 (U_{15}). The resulting $\mathcal{EL}$ version of LUBM then contains 86 TBox axioms using 43 concepts and 25 roles, and 5738 (67691) ABox axioms with 1555 (17174) individuals in instance U_1 (U_{15}). The rules are from the DReW LUBM benchmark queries and consist of ten programs P_0–P_9. They can be split into two categories: (C1) P_0–P_4 have between 2 and 5 dl-atoms but no input list, while (C2) P_5–P_9 have between 2 and 9 dl-atoms, each with distinct input list.

We used two Datalog engines to compute the models of the native and the modular encodings: clingo 3.0.3 [22] and DLV 2010-10-14 [38]. We compared TD-MLP to DReW using four benchmark run settings (the systems in square brackets denote the model builders used to calculate the model): (i) DReW[clingo], (ii) DReW[DLV], (iii) TD-MLP[clingo], and (iv) TD-MLP[DLV]. The test results are shown in Table 5a for $KB = (U_1, P_j)$ and in Table 5b for $KB = (U_{15}, P_j)$. More details are available on our benchmark webpage.[3]

DReW outperforms TD-MLP in all tests. But DReW's lead is shrinking if we increase the number of dl-atoms in the dl-programs from category (C2), i.e., (U_i, P_5)–(U_i, P_9). The reason is that with DReW we create copies of the ontology as Datalog rules for every dl-atom upfront, thus creating a large single Datalog program. In TD-MLP, we can always use a single copy of the rewritten ontology and let the MLP semantics create the copies for us. As the current TD-MLP implementation is not sophisticated enough, the overhead for instantiating modules during evaluation is prevalent.

6 Discussion and Conclusion

The experimental implementations that we used are not optimized, and in order to get a clearer picture, a more extensive experimentation is necessary. However, the results above show that the uniform evaluation approach has great potential for dl-programs, and that it outperforms a reasoner-coupling implementation substantially. Regarding the various formalisms that we considered, we make the following observations.

FO-Rewritability. The notion of FO-rewritability in Section 3 may be relaxed to permit modifications of the ABox, in spirit of the combined approach [36], where the ABox is extended dependent on the TBox but independent of the query, which is rewritten to a FO query over the extended ABox. For (acyclic) dl-programs to stay within FO, we only can have ABox extensions which do not depend on the contents of the ABox, as its precise contents in evaluating a dl-atom under updates is not known a priori. This

oarieli@mta.ac.il

still leaves the possibility for auxiliary relations that depend on the TBox and the original ABox (e.g., a linear ordering of the individuals, or arithmetic operations). Such generalized FO-rewritability increases the expressiveness in general.

By exploiting recursive query processing in SQL, uniform evaluation of dl-programs with DL-Lite$_R$ allows for performing computations like transitive closure on top of a knowledge base. However, in current RDBMS, recursive query processing is still not very advanced and has lots of room for improvement. Our experience with Postgres is in analogy to tests with commercial RDBMS (cf. [49]). A useful improvement of Postgres in this regard would be efficient cycle detection, eliminating the LIMIT parameter.[8] To cater for (also non-linear) recursion, languages like Datalog seem more attractive.

Datalog$^\neg$-Rewritability. Similar as for FO-rewritability, we also can consider here relaxed notions of Datalog-rewritability that allow for auxiliary relations. For Datalog$^\neg$ rewritability of a dl-program, we required that dl-atoms resp. the underlying description logic are Datalog rewritable. One also could consider a more liberal notion of Datalog$^\neg$ rewritability of dl-atoms, which however would need to deal with the fact that in principle, the program $\Phi_{\mathcal{DL}}(L)$ could have multiple answer sets (or none). Furthermore, simply plugging in $\Phi_{\mathcal{DL}}(L_\lambda)$ for some dl-atom $DL[\lambda, Q](\boldsymbol{t}]$ may lead to unwanted effects, due to nonmonotonicity; e.g., additional answer sets might emerge. Syntactic restrictions (e.g., acyclicity, or more general that dl-atoms are not involved in cycles) can avoid this.

Modular Logic Programs. The modular program encoding lags in total runtime behind the ad hoc inlined approach, but it scale at a slower growth rate. This looks promising as the gap for large amounts of data will be closed. In designing modules for dl-atoms, one has a range of possibilities, from few, more general modules to many but very specialized ones (in the extremal case, a single universal module for all dl-atoms (similarly as in dlvhex), vs. one module for each dl-atom). Determining the effects of such design choices, and the resulting transformations is an interesting subject fur future study.

6.1 Further Work and Outlook

The uniform evaluation approach we proposed is a flexible framework. Once a description logic has a transformation into the suitable logic, it can be easily integrated.

At the software level, in case of FO a respective component can be integrated in the MOR prototype and in case of Datalog in the DReW reasoner. In particular, (relaxed) Datalog rewritings of description logics apart from $\mathcal{EL}$ have gained attention recently, e.g. [44,24], including conjunctive query answering in Horn-$\mathcal{SHIQ}$, which subsumes all the OWL 2 profiles EL, RL and QL; an implementation is in progress [50].

We considered some formalisms for uniform evaluation of dl-programs, for which evaluation engines are available. Of course, further such formalisms (e.g., FO(·) Logic [14], or F-logic [35]) may be considered. But also formalisms for which reasoning engines are yet emerging might be of interest, e.g. Datalog $^\pm$ [7]. The latter extends Datalog with existential quantification in rule head and at the same time restricts the

[8] http://www.postgresql.org/docs/8.4/static/queries-with.html

oarieli@mta.ac.il

syntax such that reasoning remains decidable. Datalog $^{\pm}$ is more expressive than various description logics, including DL-Lite (which is captured elegantly), and allows for handling unknown individuals in the reasoning. It seems to be an attractive formalism in particular to host the combination of rules and ontologies.

Finally, the prototypes we used for experimentation are not optimized, and there is considerable room for improvement. Furthermore, alternative encodings and reductions might be considered, and specific optimization methods and techniques developed.

References

1. Acciarri, A., Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: Quonto: Querying ontologies. In: AAAI'05. pp. 1670–1671 (2005)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The dl-lite family and relations. J. Artif. Intell. Res. 36, 1–69 (2009)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: IJCAI'05. pp. 364–369. (2005)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope further. In: OWLED08-DC (2008)
5. Baader, F., Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms. J. Autom. Reasoning 14(1), 149–180 (1995)
6. Bonatti, P.A., Faella, M., Sauro, L.: Adding default attributes to EL++. In: AAAI'11. (2011)
7. Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., Pieris, A.: Datalog+/-: A family of logical knowledge representation and query languages for new applications. In: LICS'10. pp. 228–242. (2010)
8. Calvanese, D., de Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J Autom. Reasoning 39(3), 385–429 (2007)
9. Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Modular nonmonotonic logic programming revisited. In: ICLP'09. pp. 145–159. Springer (2009)
10. Dao-Tran, M., Eiter, T., Fink, M., Krennwallner, T.: Relevance-driven evaluation of modular nonmonotonic logic programs. In: LPNMR'09, pp. 87–100. Springer (2009)
11. Dao-Tran, M., Eiter, T., Krennwallner, T.: Realizing default logic over description logic knowledge bases. In: ECSQARU'09, pp. 602–613. Springer (2009)
12. de Bruijn, J., Bonnard, P., Citeau, H., Dehors, S., Heymans, S., Pührer, J., Eiter, T.: Combinations of rules and ontologies: State-of-the-art survey of issues. Tech. Rep. Ontorule D3.1, Ontorule Project Consortium (2009)
13. de Bruijn, J., Eiter, T., Tompits, H.: Embedding approaches to combining rules and ontologies into autoepistemic logic. In: KR'08, pp. 485–495 (2008)
14. Denecker, M., Ternovska, E.: A logic of non-monotone inductive definitions. ACM Trans. Comput. Log. 9(2), 14, 52pp. (2008)
15. Drabent, W., Eiter, T., Ianni, G., Krennwallner, T., Lukasiewicz, T., Małuszyński, J.: Hybrid reasoning with rules and ontologies. In: Semantic Techniques for the Web: The REWERSE perspective, chap. 1, pp. 1–49. Springer (2009)
16. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R.: Well-founded semantics for description logic programs in the Semantic Web. ACM Trans. Comput. Log. 12(2), 11, 41pp. (2011)
17. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. Artif. Intell. 172(12-13), 1495–1539 (2008)
18. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Effective integration of declarative rules with external evaluations for semantic-web reasoning. In: ESWC'06. pp. 273–287. Springer (2006)

oarieli@mta.ac.il

19. Eiter, T., Fink, M., Krennwallner, T.: Decomposition of declarative knowledge bases with external functions. In: IJCAI'09, pp. 752–758. (2009)
20. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer set programming. In: IJCAI'05, pp. 90–96. (2005)
21. Fink, M., Pearce, D.: A logical semantics for description logic programs. In: JELIA'10. pp. 156–168. Springer (2010)
22. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.: Potassco: The Potsdam Answer Set Solving Collection. AI Commun. 24(2), 107–124 (2011)
23. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385 (1991)
24. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: DL'11. CEUR-WS, vol. 745. (2011)
25. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logic. In: WWW'03. pp. 48–57. ACM (2003)
26. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for OWL knowledge base systems. J. Web Sem. 3(2-3), 158 – 182 (2005)
27. Haarslev, V., Hidde, K., Möller, R., Wessel, M.: The RacerPro knowledge representation and reasoning system. In: Semant. Web J. http://www.semantic-web-journal.net/, to appear
28. Heymans, S., Eiter, T., Xiao, G.: Tractable reasoning with DL-programs over datalog-rewritable description logics. In: ECAI'10. IOS Press (2010)
29. Heymans, S., Korf, R., Erdmann, M., Pührer, J., Eiter, T.: Loosely coupling f-logic rules and ontologies. In: WI'10, pp. 248–255. IEEE Computer Society (2010)
30. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive datalog. J. Autom. Reasoning 39(3), 351–384 (2007)
31. Janhunen, T.: On the intertranslatability of non-monotonic logics. Ann. Math. Artif. Intell. 27(1-4), 79–128 (1999)
32. Janhunen, T., Oikarinen, E., Tompits, H., Woltran, S.: Modularity Aspects of Disjunctive Stable Models. J. Artif. Intell. Res. 35, 813–857 (2009)
33. Kazakov, Y.: Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In: IJCAI'09. pp. 2040–2045 (2009)
34. Kazakov, Y., Krötzsch, M., Simancik, F.: Concurrent classification of el ontologies. In: ISWC'11, pp. 305–320. Springer (2011)
35. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. J. ACM 42(4), 741–843 (1995)
36. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in dl-lite. In: KR'10. AAAI Press (2010)
37. Krötzsch, M.: Efficient rule-based inferencing for OWL EL. In: IJCAI'11, pp. 2668–2673. (2011)
38. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV System for Knowledge Representation and Reasoning. ACM Trans. Comput. Log. 7(3) (2006)
39. Lukasiewicz, T.: A novel combination of answer set programming with description logics for the semantic web. IEEE Trans. Knowl. Data Eng. 22(11), 1577–1592 (2010)
40. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Profiles. W3C (2009), W3C Rec. 27 Oct. 2009, http://www.w3.org/TR/owl2-profiles/
41. Motik, B., Rosati, R.: Reconciling Description Logics and Rules. J. ACM 57(5), 1–62 (2010)
42. Motik, B., Shearer, R., Horrocks, I.: Hypertableau Reasoning for Description Logics. J. Artif. Intell. Res. 36, 165–228 (2009)
43. Ordonez, C.: Optimization of linear recursive queries in SQL. IEEE Trans. Knowl. Data Eng. 22(2), 264–277 (2010)
44. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the horn-DL fragments of OWL 1 and 2. In: KR'10. AAAI Press (2010)

oarieli@mta.ac.il

45. Rosati, R., Almatelli, A.: Improving query answering over dl-lite ontologies. In: KR'10. AAAI Press (2010)
46. Schneider, P.: Evaluation of description logic programs using an RDBMS. Master's thesis, TU Wien (2010)
47. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Sem. 5(2), 51–53 (2007)
48. Stocker, M., Smith, M.: Owlgres: A scalable OWL reasoner. In: OWLED'01 (2008)
49. Terracina, G., Leone, N., Lio, V., Panetta, C.: Experimenting with recursive queries in database and logic programming systems. Theor. Pract. Log. Prog. 8(2), 129–165 (2008)
50. Tran, T.K.: Query answering in the description logic Horn-SHIQ. Master's thesis, TU Wien (2011)
51. Wang, K., Billington, D., Blee, J., Antoniou, G.: Combining description logic and defeasible logic for the semantic web. In: RuleML'04, pp. 170–181. Springer (2004)
52. Wang, Y., You, J.H., Yuan, L.Y., Shen, Y.D.: Loop formulas for description logic programs. Theor. Pract. Log. Prog. 10(4-6), 531–545 (2010)
53. Wang, Y., You, J.H., Yuan, L.Y., Shen, Y.D., Eiter, T.: Embedding description logic programs into default logic. CoRR abs/1111.1486 (2011)
54. Wijaya, T.: Top-Down Evaluation Techniques for Modular Nonmonotonic Logic Programs. Master's thesis, TU Wien (2011), http://media.obvsg.at/AC07811177-2001
55. Xiao, G., Eiter, T.: Inline evaluation of hybrid knowledge bases – PhD description. In: RR'11, pp. 300–305. Springer (2011)
56. Xiao, G., Heymans, S., Eiter, T.: DReW: a reasoner for datalog-rewritable description logics and dl-programs. In: BuRO'10 (2010), http://ontorule-project.eu/attachments/075_buro2010-proceedings.pdf

oarieli@mta.ac.il

Foundations of XML Based on Logic and Automata: A Snapshot

Thomas Schwentick*

Technische Universität Dortmund

Abstract. XML query and schema languages have some obvious connections to Formal Language Theory. For example, Document Type Definitions (DTDs) can be viewed as tree grammars and use regular expressions, XML Schemas resemble tree automata. Likewise, there are immediate links to Logic, e.g., through the classical characterization of regular tree languages by monadic second-order logic.

It is therefore not surprising that concepts from Logic and Formal Language Theory played an important role in the development of the theoretical foundations of XML query and schema languages. For example, they helped to analyze the expressiveness of languages, to understand the restrictions posed by the W3C standards, and to develop algorithms for various purposes, including efficient evaluation and static analysis.

However, methods from Logic and Formal Languages have not merely been applied to XML theory, the fertilization took place both ways. XML theory posed a lot of new challenges for Logic and Formal Language Theory and triggered various new research lines, e.g., the study of deterministic regular expressions and the development of automata for trees with data values.

The aim of the talk at FoIKS 2012 is to present some of the fundamental connections between XML query and schema languages and Logic and Formal Language Theory, to report on recent developments in the area, and to highlight some current directions of research. This accompanying paper is a kind of annotated bibliography for that talk.

The Extensible Markup Language (XML) was introduced in 1996. It was derived as a "simple dialect of SGML" with the purpose "to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML" [26]. Meanwhile XML is the standard data format for the exchange of data through the internet. For the current standard of XML 1.0, see [27].

For most theoretical purposes, XML documents can be adequately modelled as finite unranked, ordered, labelled trees with additional data values. XML query and schema languages and many other aspects of XML processing have some obvious connections to Formal Language Theory (henceforth, *FLT*) and Logics.

* We acknowledge the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under the FET-Open grant agreement FOX, number FP7-ICT-233599.

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 23–33, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

The very first such connection is given by the common requirement that XML documents that are used in a certain context should be *valid* with respect to a given *document type definition (DTD)*. Valid XML documents can be seen as derivation trees with respect to an extended context-free grammar [7]. In formal language terminology, the set of valid documents for a given DTD is just a *tree language*.

DTD is "the built-in" *schema language* of the XML standard but there are various stronger schema languages, most notably XML Schema [38], the W3C standard, and Relax NG [33]. The sets of XML documents that can be described with them correspond to *regular tree languages* and therefore are strongly related to *finite tree automata*. A classification of XML schema languages from the point of view of FLT has been given in [73].

Here, we can already see a pattern that frequently occurs in theoretical XML research: a concept in the XML world (schema languages) is modelled by a concept from FLT (regular tree languages). However, to satisfy the needs of the XML side, the classical FLT notion has to be adapted: classically, the labels of nodes in regular tree languages come with a fixed rank. E.g., the label a of a node v might indicate that v is a binary node, that is, it has two children nodes. In particular, in trees of such languages there is a uniform bound on the number of children nodes that a node may have. In XML documents, there is no such (a priori) bound, therefore the theory of regular languages had to be extended to regular languages of *unranked* trees. To this end, tree automata have been invented that can handle trees languages without a rank bound [72,30,74] (see also [81] for a survey on automata models for XML).

This paper is about the interplay between XML Theory on one hand and FLT and Logic on the other hand. We will first survey applications of FLT and Logic in XML Theory. We will see examples highlighting

- how the expressiveness of XML languages can be analyzed and characterized by logics,
- how Formal Language Theory can help to understand restrictions posed by W3C standards, and
- how concepts from FLT can be used to device evaluation and static analysis algorithms for various XML languages.

However, the relationship between XML Theory, FLT and Logic is not one-way. Investigations in XML Theory brought up new challenges for the latter and we will sketch some cases where such challenges triggered new research lines in FLT and Logic themselves.

Due to a bounded time budget, this paper can only highlight these connections. By no means it is meant as a complete account. Furthermore, it is biased as it is strongly influenced by the own research of the author and his closer colleagues. I apologize for ignoring important papers and even complete lines of research. Another shortcoming of the paper is that it is by no means self contained. It does not even define the most simple notions in any detail. It is rather meant as an annotated bibliography in that respect. For technical details we refer to the original papers or other, more in-depth surveys, like [75,76,59,83,84,81,68].

oarieli@mta.ac.il

Finally it should be noted that citations in tis paper usually refer to the longer journal version of a result, sometimes distorting the order of discoveries.

1 Using Formal Language Theory and Logic for XML Theory

1.1 Analyzing Expressive Power

We first consider characterizations of the expressiveness of XML languages, that is, results of the kind "a query can be stated in query language $\mathcal{Q}$ if and only if it can be expressed in logic $\mathcal{L}$". Of course, such results always refer to reasonably abstracted versions and precisely defined subsets of these query or schema languages, leaving out some bells and whistles. As a matter of fact, such characterizations exist for all main standard languages for validation (XML Schema, [38]), navigation, (XPath, [36,37]), querying (XQuery, [32]) and tranformation (XSLT, [34]) of XML documents.

One of the first correspondences between Logic and XML languages is through the classical logic-automata connection for regular tree languages. It has been known for more than 50 years that regular string languages and thus finite string automata correspond to monadic second-order logic (MSO), the extension of first-order logic by set quantification, [31,42,91]. The same correspondence holds for regular languages of finite ranked trees [41,89] and it carries over in a straightforward manner to regular languages of unranked trees [77]. For more information on regular languages of ranked trees and monadic second-order logic we refer to [90]. The already mentioned correspondence between XML schema languages and regular languages of unranked trees thus yields a correspondence between schema languages and MSO logic. In fact, the language Relax NG (see relaxng.org and [33]) was specifically designed to allow the full power of regular tree languages in a schema language (and RELAX is just an acronym for *Regular Language description for XML*). However, XML Schema does not capture all[1] regular tree languages but it is still useful that they can be investigated within the framework of regular tree languages.

Very close correspondences with Logic have been established for XML navigation languages. In [66] (see also [69]), it was shown that a node-selecting query can be stated in the navigational core of XPath 1.0 [36] if and only if it can be expressed in two-variable logic, the restriction of first-order logic to formulas with two variables. XPath 2.0 [37] however corresponds to first-order logic [67] (see also [87]). In fact, some of its extensions were explicitly added to obtain the full expressive power of first-order logic [87].

The relationship between the XML query language XQuery and Logic has been studied in [5,6]. It turns out that XQuery queries that do not use *deep equality* can be captured by first-order interpretations (mappings defined by first-order formulas with free variables). Queries with deep equality can be captured

[1] A precise characterization of the subclass of regular tree languages that can be specified in XML Schema is discussed further below.

oarieli@mta.ac.il

by the counting extension of first-order logic. These correspondences do not yield complete characterizations, as not all first-order interpretations can be decribed in XQuery. However, a large class of first-order interpretations can also be translated back into XQuery [5].

The relationship between the XML transformation language XSLT and FLT and Logic is more complicated. Large parts of XSLT can be captured by various models of tree-walking transducers with pebbles [70,8,44] and macro tree transducers [60]. Restrictions of such transducers in turn closely correspond to MSO logic [70,45]. We also refer to [71].

1.2 Understanding Restrictions

As mentioned before, DTDs are essentially extended context-free grammars. The term *extended* refers to the fact that the right-hand sides of productions can be regular expressions rather than strings of variables and terminals as in standard grammars. However, these regular expressions have to be *1-unambiguous* (as already required in SGML), meaning that a scanner can always uniquely match a stream of symbols with the symbols of the expression without look-ahead. In XML Schema, a similar restriction is given by the *Unique Particle Attribution (UPA)* constraint. This constraint can be understood much easier with the help of automata. There is a straightforward way to translate regular expressions into (possibly nondeterministic) finite state automata given by Glushkov [53]. If the resulting automaton is deterministic then the regular expression fulfills the constraint [28]. Therefore, these expressions are more often called *deterministic regular expressions*.

It is easy to decide whether a given regular expression is deterministic. However, not every regular language can be described by a deterministic regular expression and it is less obvious how to check whether this is the case for a given language (say, represented by a finite automaton). Nevertheless, it was shown in [28] that this can actually be decided in quadratic time by a non-trivial inspection of the minimal automaton of the language.

A second important constraint of XML Schema is the *Element Declaration Consistent (EDC)* constraint. In [73,65] this constraint was characterized in terms of Formal Language Theory. It yields a strict subclass of the regular tree languages. Further charaterizations were given in [65]. In fact, a language respects this constraint if and only if it has a very simple closure property. Furthermore, the characterization yields a simple pattern based language with the same expressive power as XML Schema (see also [63]).

1.3 Algorithms Based on Formal Language Theory

In [55] it was pointed out that XPath processors available at the time show anexponential time worst-case behavior when evaluating XPath queries. The paper gives a dynamic programming based polynomial time algorithm for the evaluation of XPath and a linear time (both in the query and the document size) algorithm for the core (navigational) fragment of XPath. Surprisingly, it

oarieli@mta.ac.il

was shown in [21,22] that linear time algorithms (in the size of the document) even exist for a larger fragment of XPath with node tests with attribute and text values. These algorithms make sophisticated use of automata.

Another area where automata are used as an algorithmic tool is static analysis of XML queries and schemas. Examples are algorithms for

- testing satisfiability of XPath expressions [4],
- testing containment of XPath expressions [79,86],
- type checking of XSLT transformations [70,60,61,62],
- analyzing XML Schemas [65].

2 Challenges for Formal Language Theory and Logic

However, methods from Logic and Formal Languages have not merely been applied to XML theory, the fertilization took place in both directions. The specifics of XML posed a lot of new challenges for Logic and Formal Language Theory and triggered various new research lines. In the following, we highlight three such lines of research.

2.1 Regular Expression Research Triggered by XML

We already mentioned the decision algorithm that checks whether a given regular language can be expressed by a deterministic regular expression [28] and described the relevance of deterministic regular expressions for XML schema languages. The need to deal with XML schemas and the aim to understand XML schema languages initiated many more investigations on (deterministic) regular expressions.

The class of languages described by deterministic regular expressions is not closed under union, intersection or complementation [29,52]. As the ability to intersect and merge content models is crucial for the manipulation of XML schemas, [9] studied approximations of regular languages by deterministic regular expression. In [52] the increase of the size of regular expressions under complementation and intersection is studied. It is shown that the minimum size of a regular expression for the complement language of a given regular expression can be doubly-exponential, for the intersection of two regular expressions it can be exponential. The latter lower bound even holds if the two given expressions are deterministic. However, a (not necessarily deterministic) regular expression for the complement of a language given by a deterministic regular expression can be computed in polynomial time.

Most regular expressions used in real-life schemas are of a simple structure [50]. This motivates the definition and study of fragments of regular expressions with better algorithmic properties [64]. On the other hand, regular expressions with additional syntatctic sugar, e.g., with `minoccur` and `maxoccur` bounds, need to be investigated as well [51].

The inference problem for XML schemas, that is, the problem to infer a schema from a given set of XML documents, basically boils down to learning

oarieli@mta.ac.il

(deterministic) regular expressions and has triggered many new investigations in algorithmic learning [49,12,13,10,11].

2.2 Data Trees and Data Words

Most of the investigations mentioned so far concentrate on the structural aspects of XML documents. That is, they basically ignore attribute and text values and consider XML documents as trees that are labelled by tags from a finite set of tags (possibly induced by the schema). Whereas this is a reasonable abstraction for many purposes, obviously not all aspects of XML processing can be studied with this model. A more versatile abstraction needs to take attribute and text values into account.

In recent years, the model of *data trees* was invented and its properties were investigated from different angles, in particular from the point of view of Logic and FLT [20,14,19]. In a data tree, nodes carry a label from a finite alphabet and a value from a possible infinite domain. In this way, XML tags can still be modelled by the label, but attribute and text values can be represented by the infinite domain value. The model builds on earlier work on languages over "infinite alphabets" [57,25].

In a very nice series of papers, the static analysis of XPath (with node tests on attribute and text values) has been studied and the frontier of decidability of the satisfiability problem for XPath expressions, based on the allowed axes, has been largely explored. Most of these results were shown with the help of different kinds of automata models for data trees. Whereas satisfiability of XPath queries is undecidable in general [4], it is decidable if only forward axes [47] or vertical axes [48] are used. If only downward axes are allowed it even has reasonable complexity [46].

Interestingly, the corresponding extension of strings (over finite alphabets) by values from an infinite domain, that is *data strings*, is also intensely studied in the context of automatic verification [39,24,40]. Thus, in this case the cross-fertilization involves a third partner, automated verification. Many questions about automata and logics on data words and data trees are wide open. As an example, it is unclear whether there exist robust classes of languages that can be seen as counter parts of regular string and tree languages [16]. Currently, the development of an algebraic theory for data languages is a very active research area [17,35].

2.3 Pebble Automata and Transitive-Closure Logic on Trees

Regular tree languages are defined with the help of automata that traverse the tree in parallel, e.g., from the leaves of the tree to its root. As mentioned above, they correspond to monadic second-order logic. Some aspects of XML processing are rather of a sequential nature, that is, the tree is processed one node after the other. These aspects can be modelled by other, more sequential, types of automata. The basic model of sequential automata on trees is given by *tree-walking automata* [1]. They are equipped with a single head that moves along the tree. *Pebble automata*

oarieli@mta.ac.il

extend tree-walking automata with pebbles, these are essentially node markers that the head can put at and take from a node but only in a LIFO fashion. The relationship between tree-walking automata and XPath is dicussed in [54]. Pebble automata have also been used to model XML transformations, e.g. in [70]. The expressive power of sequential automata models, in particular whether they are able to capture all regular tree languages and thus MSO logic, has been open for many years. This question was raised already in [80] and was repeated in other papers [78,43]. In [43], it was shown that pebble automata correspond to transitive closure logic on trees where (1) the transitive closure operator can only be applied to binary relations and (2) it is not allowed to occur in the scope of negation. Thus, the question about the strength of sequential automata is strongly related to the purely logical question how unary transitive closure logic compares to monadic second-order logic on trees. We note that these two logics have exactly the same expressive power on strings.

The question was resolved first for tree-walking automata in [18]: they can not express all regular tree languages. Building on that work, it has been shown in [23] that tree-walking automata with pebbles also do *not* capture the regular tree languages and thus that the transitive closure logic from [43] is weaker than MSO logic. Finally, it has been shown in [88] that actually the full unary transitive closure logic (that is, with restriction (1) but without restriction (2)) is weaker than MSO logic.

3 Conclusion

As mentioned in the introduction, this paper does not give by any means a full account on the relationships between XML Theory, Formal Language Theory and Logic. Many topics have not been mentioned at all, like incremental validation of XML documents [2,3] or incremental evaluation of XPath queries [15], stream-based processing [56,58,85] and many others.

Even though XML has been around for more than 15 years, it remains a very active research area. And it has initiated many research lines in other areas, some of which have been pointed out above.

References

1. Aho, A.V., Ullman, J.D.: Translations on a context-free grammar. Information and Control 19(5), 439–475 (1971)
2. Balmin, A., Papakonstantinou, Y., Vianu, V.: Incremental validation of xml documents. ACM Trans. Database Syst. 29(4), 710–751 (2004)
3. Barbosa, D., Mendelzon, A.O., Libkin, L., Mignet, L., Arenas, M.: Efficient incremental validation of XML documents. In: ICDE, pp. 671–682. IEEE Computer Society (2004)
4. Benedikt, M., Fan, W., Geerts, F.: XPath satisfiability in the presence of DTDs. J. ACM 55(2) (2008)
5. Benedikt, M., Koch, C.: Interpreting Tree-to-Tree Queries. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 552–564. Springer, Heidelberg (2006)

oarieli@mta.ac.il

6. Benedikt, M., Koch, C.: From XQuery to relational logics. ACM Trans. Database Syst. 34(4) (2009)
7. Berstel, J., Boasson, L.: Formal properties of XML grammars and languages. Acta Inf. 38(9), 649–671 (2002)
8. Bex, G.J., Maneth, S., Neven, F.: A Formal Model for an Expressive Fragment of XSLT. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 1137–1151. Springer, Heidelberg (2000)
9. Bex, G.J., Gelade, W., Martens, W., Neven, F.: Simplifying xml schema: effortless handling of nondeterministic regular expressions. In: Çetintemel, U., Zdonik, S.B., Kossmann, D., Tatbul, N. (eds.) SIGMOD Conference, pp. 731–744. ACM (2009)
10. Bex, G.J., Gelade, W., Neven, F., Vansummeren, S.: Learning deterministic regular expressions for the inference of schemas from XML data. TWEB 4(4) (2010)
11. Bex, G.J., Neven, F., Schwentick, T., Vansummeren, S.: Inference of concise regular expressions and DTDs. ACM Trans. Database Syst. 35(2) (2010)
12. Bex, G.J., Neven, F., Vansummeren, S.: Inferring XML schema definitions from XML data. In: VLDB, pp. 998–1009 (2007)
13. Bex, G.J., Neven, F., Vansummeren, S.: Schemascope: a system for inferring and cleaning xml schemas. In: Wang, J.T.-L. (ed.) SIGMOD Conference, pp. 1259–1262. ACM (2008)
14. Björklund, H., Bojańczyk, M.: Bounded Depth Data Trees. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 862–874. Springer, Heidelberg (2007)
15. Björklund, H., Gelade, W., Martens, W.: Incremental xpath evaluation. ACM Trans. Database Syst. 35(4), 29 (2010)
16. Björklund, H., Schwentick, T.: On notions of regularity for data languages. Theor. Comput. Sci. 411(4-5), 702–715 (2010)
17. Bojanczyk, M.: Data monoids. In: Schwentick, Dürr [82], pp. 105–116
18. Bojanczyk, M., Colcombet, T.: Tree-walking automata do not recognize all regular languages. SIAM J. Comput. 38(2), 658–701 (2008)
19. Bojanczyk, M., Lasota, S.: An extension of data automata that captures xpath. In: LICS, pp. 243–252. IEEE Computer Society (2010)
20. Bojanczyk, M., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data trees and xml reasoning. J. ACM 56(3) (2009)
21. Bojanczyk, M., Parys, P.: XPath evaluation in linear time. In: Lenzerini, M., Lembo, D. (eds.) PODS, pp. 241–250. ACM (2008)
22. Bojanczyk, M., Parys, P.: XPath evaluation in linear time. J. ACM 58(4), 17 (2011)
23. Bojańczyk, M., Samuelides, M., Schwentick, T., Segoufin, L.: Expressive Power of Pebble Automata. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part I. LNCS, vol. 4051, pp. 157–168. Springer, Heidelberg (2006)
24. Bouajjani, A., Habermehl, P., Jurski, Y., Sighireanu, M.: Rewriting Systems with Data. In: Csuhaj-Varjú, E., Ésik, Z. (eds.) FCT 2007. LNCS, vol. 4639, pp. 1–22. Springer, Heidelberg (2007)
25. Bouyer, P., Petit, A., Therien, D.: An algebraic approach to data languages and timed languages. Inf. Comput. 182(2), 137–162 (2003)
26. Bray, T., Paoli, J., Sperberg-McQueen, C.M.: Extensible markup language (XML) working draft. Technical report, W3C (1996), www.w3.org/TR/WD-xml-961114
27. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible markup language (XML) 1.0 (fifth edition). Technical report, W3C (2008), www.w3.org/TR/2008/REC-xml-20081126

oarieli@mta.ac.il

28. Brüggemann-Klein, A., Wood, D.: Deterministic regular languages. Technical Report Bericht 38, Universität Freiburg, Institut für Informatik (1991)
29. Brüggemann-Klein, A., Wood, D.: One-unambiguous regular languages. Information and Computation 140(2), 229–253 (1998)
30. Brüggemann-Klein, A., Murata, M., Wood, D.: Regular tree languages over non-ranked alphabets (1998), http://hdl.handle.net/1783.1/738
31. Büchi, J.R.: Weak second-order arithmetic and finite automata. Z. Math. Logik Grundl. Math. 6, 66–92 (1960)
32. Chamberlin, D., Clark, J., Florescu, D., Robie, J., Simeon, J., Stefanascu, M.: XQuery 1.0: An XML query language (2002), http://www.w3.org/TR/xquery/
33. Clark, J., Murata, M.: Relax ng specification. Technical report, OASIS Committee Specification Document (2001)
34. Clark, J.: XSL transformations version 1.0 (August 1999), http://www.w3.org/TR/xslt
35. Colcombet, T., Ley, C., Puppis, G.: On the Use of Guards for Logics with Data. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 243–255. Springer, Heidelberg (2011)
36. World Wide Web Consortium. XML Path Language (XPath), Version 1.0. W3C Recommendation, November 16 (1999), http://www.w3.org/TR/xpath
37. World Wide Web Consortium. XML Path Language (XPath), Version 2.0. W3C Recommendation, http://www.w3.org/TR/xpath20
38. World Wide Web Consortium. XML schema, http://www.w3.org/XML/Schema
39. Demri, S., Lazic, R.: LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log. 10(3) (2009)
40. Demri, S., Sangnier, A.: When Model-Checking Freeze ltl Over Counter Machines Becomes Decidable. In: Ong, L. (ed.) FOSSACS 2010. LNCS, vol. 6014, pp. 176–190. Springer, Heidelberg (2010)
41. Doner, J.: Tree acceptors and some of their applications. J. Comput. Syst. Sci. 4(5), 406–451 (1970)
42. Elgot, C.C.: Decision problems of finite automata design and related arithmetics. Transactions of The American Mathematical Society 98, 21–21 (1961)
43. Engelfriet, J., Hoogeboom, H.J.: Automata with nested pebbles capture first-order logic with transitive closure. Logical Methods in Computer Science 3(2) (2007)
44. Engelfriet, J., Hoogeboom, H.J., Samwel, B.: Xml transformation by tree-walking transducers with invisible pebbles. In: Libkin, L. (ed.) PODS, pp. 63–72. ACM (2007)
45. Engelfriet, J., Maneth, S.: Macro tree transducers, attribute grammars, and mso definable tree translations. Inf. Comput. 154(1), 34–91 (1999)
46. Figueira, D.: Satisfiability of downward XPath with data equality tests. In: PODS, pp. 197–206 (2009)
47. Figueira, D.: Forward-XPath and extended register automata on data-trees. In: ICDT, pp. 231–241 (2010)
48. Figueira, D., Segoufin, L.: Bottom-up automata on data trees and vertical xpath. In: Schwentick, Dürr [82], pp. 93–104
49. Garofalakis, M.N., Gionis, A., Rastogi, R., Seshadri, S., Shim, K.: Xtract: Learning document type descriptors from xml document collections. Data Min. Knowl. Discov. 7(1), 23–56 (2003)
50. Van den Bussche, J., Bex, G.J., Neven, F.: DTDs versus XML schema: A practical study. In: International Workshop on the World Wide Web and Databases, pp. 79–84 (2004)

oarieli@mta.ac.il

51. Gelade, W., Gyssens, M., Martens, W.: Regular Expressions with Counting: Weak versus Strong Determinism. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 369–381. Springer, Heidelberg (2009)
52. Gelade, W., Neven, F.: Succinctness of the complement and intersection of regular expressions. In: Albers, S., Weil, P. (eds.) STACS. LIPIcs, vol. 1, pp. 325–336. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2008)
53. Glushkov, V.M.: The abstract theory of automata. Russian Math. Surveys 16, 1–53 (1961)
54. Goris, E., Marx, M.: Looping caterpillars. In: LICS 2005 (2005)
55. Gottlob, G., Koch, C., Pichler, R.: Efficient algorithms for processing xpath queries. ACM Trans. Database Syst. 30(2), 444–491 (2005)
56. Green, T.J., Miklau, G., Onizuka, M., Suciu, D.: Processing XML Streams with Deterministic Automata. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 173–189. Springer, Heidelberg (2002)
57. Kaminski, M., Francez, N.: Finite-memory automata. Theor. Comput. Sci. 134(2), 329–363 (1994)
58. Koch, C., Scherzinger, S., Schweikardt, N., Stegmaier, B.: FluXQuery: An optimizing XQuery processor for streaming XML data. In: Proc. 30th International Conference on Very Large Data Bases (VLDB), Toronto, pp. 1309–1312 (2004)
59. Libkin, L.: Logics for unranked trees: An overview. Logical Methods in Computer Science 2(3) (2006)
60. Maneth, S., Berlea, A., Perst, T., Seidl, H.: XML type checking with macro tree transducers. In: PODS, pp. 283–294 (2005)
61. Maneth, S., Perst, T., Seidl, H.: Exact XML Type Checking in Polynomial Time. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 254–268. Springer, Heidelberg (2006)
62. Martens, W., Neven, F.: Frontiers of tractability for typechecking simple xml transformations. J. Comput. Syst. Sci. 73(3), 362–390 (2007)
63. Martens, W., Neven, F., Schwentick, T.: Simple off the shelf abstractions for XML schema. SIGMOD Record 36(3), 15–22 (2007)
64. Martens, W., Neven, F., Schwentick, T.: Complexity of decision problems for XML schemas and chain regular expressions. SIAM J. Comput. 39(4), 1486–1530 (2009)
65. Martens, W., Neven, F., Schwentick, T., Bex, G.J.: Expressiveness and complexity of XML schema. ACM Trans. Database Syst. 31(3), 770–813 (2006)
66. Marx, M., de Rijke, M.: Semantic characterizations of XPath. In: TDM 2004 Workshop on XML Databases and Information Retrieval, Twente, The Netherlands, June 21 (2004)
67. Marx, M.: Conditional xpath. ACM Trans. Database Syst. 30(4), 929–959 (2005)
68. Marx, M.: Logical Foundations of XML and XQuery. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web 2009. LNCS, vol. 5689, pp. 111–157. Springer, Heidelberg (2009)
69. Marx, M., de Rijke, M.: Semantic characterizations of navigational xpath. SIGMOD Record 34(2), 41–46 (2005)
70. Milo, T., Suciu, D., Vianu, V.: Typechecking for xml transformers. J. Comput. Syst. Sci. 66(1), 66–97 (2003)
71. Møller, A., Schwartzbach, M.I.: The Design Space of Type Checkers for XML Transformation Languages. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 17–36. Springer, Heidelberg (2005)
72. Murata, M.: Forest-regular languages and tree-regular languages. Technical report, Fuji Xerox, Japan. Technical Report (1995)

oarieli@mta.ac.il

73. Murata, M., Lee, D., Mani, M., Kawaguchi, K.: Taxonomy of XML schema languages using formal language theory. ACM Trans. Internet Techn. 5(4), 660–704 (2005)
74. Neven, F., Schwentick, T.: Query automata over finite trees. Theoretical Computer Science 275, 633–674 (2002)
75. Neven, F.: Automata, logic, and XML. In: CSL, pp. 2–26 (2002)
76. Neven, F.: Automata theory for XML researchers. SIGMOD Record 31(3), 39–46 (2002)
77. Neven, F., Schwentick, T.: Query automata over finite trees. Theoretical Computer Science 275(1-2), 633–674 (2002)
78. Neven, F., Schwentick, T.: On the power of tree-walking automata. Information and Control 183(1), 86–103 (2003)
79. Neven, F., Schwentick, T.: On the complexity of XPath containment in the presence of disjunction, DTDs, and variables. Logical Methods in Computer Science 2(3) (2006)
80. Potthoff, A.: Logische Klassifizierung regulärer Baumsprachen. Doctor's thesis, Institut für Informatik u. Prakt. Math., Universität Kiel (1994)
81. Schwentick, T.: Automata for XML - a survey. J. Comput. Syst. Sci. 73(3), 289–315 (2007)
82. Schwentick, T., Dürr, C. (eds.): 28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, Dortmund, Germany, March 10-12. LIPIcs, vol. 9. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
83. Segoufin, L.: Automata and Logics for Words and Trees over an Infinite Alphabet. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 41–57. Springer, Heidelberg (2006)
84. Segoufin, L.: Static analysis of xml processing with data values. SIGMOD Record 36(1), 31–38 (2007)
85. Segoufin, L., Vianu, V.: Validating streaming XML documents. In: PODS 2002, pp. 53–64 (2002)
86. ten Cate, B., Lutz, C.: The complexity of query containment in expressive fragments of xpath 2.0. J. ACM 56(6) (2009)
87. ten Cate, B., Marx, M.: Axiomatizing the logical core of xpath 2.0. Theory Comput. Syst. 44(4), 561–589 (2009)
88. ten Cate, B., Segoufin, L.: Transitive closure logic, nested tree walking automata, and xpath. J. ACM 57(3) (2010)
89. Thatcher, J.W., Wright, J.B.: Generalized finite automata theory with an application to a decision problem of second-order logic. Mathematical Systems Theory 2(1), 57–81 (1968)
90. Thomas, W.: Languages, Automata, and Logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. III, pp. 389–455. Springer, Heidelberg (1997)
91. Trakhtenbrot, B.A.: Finite automata and logic of monadic predicates. Doklady Akademii Nauk SSSR 140, 326–329 (1961)

oarieli@mta.ac.il

Inconsistency-Tolerance in Knowledge-Based Systems by Dissimilarities

Ofer Arieli[1] and Anna Zamansky[2]

[1] Department of Computer Science, The Academic College of Tel-Aviv, Israel
oarieli@mta.ac.il
[2] Institute for Discrete Mathematics, Vienna Technical University, Austria
annaz@logic.at

Abstract. Distance-based reasoning is a well-known approach for defining non-monotonic and paraconsistent formalisms, which so far has been mainly used in the context of standard two-valued semantics. In this paper, we extend this approach to *arbitrary denotational semantics* by considering *dissimilarities*, a generalization of distances. Dissimilarity-based reasoning is then applied for handling inconsistency in knowledge-based systems using various non-classical logics. This includes logics defined by multi-valued semantics, non-deterministic semantics, and possible-worlds (Kripke-style) semantics. In particular, we show that our approach allows to define a variety of inconsistency-tolerant entailment relations, and that it extends many well-studied forms of reasoning in the context of belief revision and database integration.

1 Introduction

A common, model-theoretic way, of defining a consequence relation for a logical system S, is to require that every model of the premises would also be a model of the conclusion. Symbolically, this can be represented as follows:

$$\Gamma \vdash_{\mathsf{S}} \psi \text{ if } mod_{\mathsf{S}}(\Gamma) \subseteq mod_{\mathsf{S}}(\psi). \tag{1}$$

Logics that are based on this approach (including, e.g., classical logic, intuitionistic logic, and many forms of modal logics) face difficulties in handling inconsistent information, since, by (1), if Γ has no model it entails *any* conclusion. This problem has long been identified, and different solutions have been proposed to it. However, many of those solutions depend on the nature of the semantics at hand, and therefore they cannot be easily adapted to other contexts.

One way of properly maintaining inconsistency, and still being faithful to (1), is to substitute $mod_{\mathsf{S}}(\Gamma)$ in (1) by a nonempty set $\Delta_{\mathsf{S}}(\Gamma)$ that coincides with $mod_{\mathsf{S}}(\Gamma)$ whenever the latter is non-empty. Intuitively, $\Delta_{\mathsf{S}}(\Gamma)$ consists of the semantic elements that are 'as close as possible' to satisfying Γ. This intuition motivates the dissimilarity-based approach for defining $\Delta_{\mathsf{S}}(\Gamma)$, introduced in [6]. The main goal of this paper is to generalize this approach to *any* denotational semantics. To this end, we define in precise terms what a dissimilarity between

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 34–50, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

semantic objects in a given denotational semantics is, and what properties it should satisfy in order to induce natural and useful entailments. This allows us to apply these abstract definitions, in a uniform way, on different kinds of denotational semantics, and not define them from scratch for each new type of semantics (as is done in [6]).

Next, given a logic L that is based on a denotational semantics, we provide a general way of constructing a logic L' that is an *inconsistency-tolerant* variant of L, in the sense that L' coincides with L with respect to consistent premises, and is non-trivial with respect to inconsistent premises. A major advantage of this approach is its uniformity: in order to construct an inconsistency-tolerant variant of one's favorite logic, one only needs to define a dissimilarity relation in this logic, and this induces a corresponding inconsistency-tolerant entailment. This approach may be useful, for instance, for applying distance-based strategies for revising or merging knowledge-bases, the semantics of which is not the standard classical one.

The rest of this paper is organized as follows: In the next section we recall some basic definitions and facts about denotational semantics and the logics induced by them. Then, in Section 3, we explain what we mean by 'inconsistency-tolerant' logics. In Section 4 we introduce dissimilarity-based semantic settings and show how they can be used for turning different types of logics, induced by denotational semantics, to inconsistency-tolerant ones. In Section 5 we examine some common properties of the entailment relations defined in Section 4, and in Section 6 we apply our framework to particular cases of denotational semantics. In Section 7 we conclude and consider some directions for future work.

2 Denotational Semantics and Their Logics

In the sequel, $\mathcal{L}$ denotes a propositional language with a countable set $\mathsf{Atoms} = \{p, q, r \ldots\}$ of atomic formulas and a (countable) set $\mathcal{F}_{\mathcal{L}} = \{\psi, \phi, \sigma, \ldots\}$ of well-formed formulas. A theory Γ is a finite set of formulas in $\mathcal{F}_{\mathcal{L}}$. The atoms appearing in the formulas of Γ and the subformulas of Γ are denoted, respectively, by $\mathsf{Atoms}(\Gamma)$ and $\mathsf{SF}(\Gamma)$. The set of all theories of $\mathcal{L}$ is denoted by $\mathcal{T}_{\mathcal{L}}$.

Definition 1. Given a language $\mathcal{L}$, a *propositional logic* for $\mathcal{L}$ is a pair $\langle \mathcal{L}, \vdash \rangle$, where $\vdash$ is a (Tarskian) consequence relation for $\mathcal{L}$, i.e., a binary relation between sets of formulas in $\mathcal{F}_{\mathcal{L}}$ and formulas in $\mathcal{F}_{\mathcal{L}}$, satisfying the following conditions:

Reflexivity: if $\psi \in \Gamma$ then $\Gamma \vdash \psi$.
Monotonicity: if $\Gamma \vdash \psi$ and $\Gamma \subseteq \Gamma'$, then $\Gamma' \vdash \psi$.
Transitivity: if $\Gamma \vdash \psi$ and $\Gamma', \psi \vdash \varphi$ then $\Gamma, \Gamma' \vdash \varphi$.

A common (model-theoretical) way of defining logics for $\mathcal{L}$ is based on the notion of *denotational semantics*:

Definition 2. A *denotational semantics* for a language $\mathcal{L}$ is a pair $\mathsf{S} = \langle S, \models_S \rangle$, where S is a nonempty set (of 'interpretations'), and $\models_S$ (the 'satisfiability relation' of S) is a binary relation on $S \times \mathcal{F}_{\mathcal{L}}$.

oarieli@mta.ac.il

Example 1. The most common case of denotational semantics is classical logic, in which the elements of S are (two-valued) *valuations*, i.e., functions from the well-formed formulas of a standard propositional language to the set $\{t, f\}$ of the classical truth values, and $\models_S$ is the ordinary satisfaction relation, defined by $\nu \models_S \Gamma$ iff $\nu(\psi) = t$ for every $\psi \in \Gamma$.

Standard generalizations of classical logic to multiple-valued logics can also be described in terms of denotational semantics and so are, e.g., the various kinds of Kripke-structures for modal and for intuitionistic logics (some of which are described in greater detail in Section 6 below).

Let $\nu \in S$ be an interpretation and $\psi \in \mathcal{F}_{\mathcal{L}}$ a formula. If $\nu \models_S \psi$, we say that ν *satisfies* ψ and call ν an S-*model* of ψ. The set of the S-models of ψ is denoted by $mod_{\mathsf{S}}(\psi)$. If $mod_{\mathsf{S}}(\psi) = S$ then ψ is called an S-*tautology*, and if $mod_{\mathsf{S}}(\psi) = \emptyset$ then ψ is called an S-*contradiction*. If ν satisfies every formula ψ in a theory Γ, it is called an S-model of Γ. The set of the S-models of Γ is denoted by $mod_{\mathsf{S}}(\Gamma)$. If $mod_{\mathsf{S}}(\Gamma) \neq \emptyset$ we say that Γ is S-*consistent*, otherwise Γ is S-*inconsistent*. In what follows we shall sometimes omit the prefix S from the above notions.

Definition 3. A denotational semantics $\mathsf{S} = \langle S, \models_S \rangle$ is *normal*, if for each $\nu \in S$ there is a formula ψ, such that $\nu \not\models_S \psi$.

Example 2. Any denotational semantics S for which there is an S-contradiction, is normal.

A denotational semantics S induces the following relation on $\mathcal{T}_{\mathcal{L}} \times \mathcal{F}_{\mathcal{L}}$:

Definition 4. $\Gamma \vdash_{\mathsf{S}} \psi$ if $mod_{\mathsf{S}}(\Gamma) \subseteq mod_{\mathsf{S}}(\psi)$.

The following proposition is easily verified.[1]

Proposition 1. *Let* $\mathsf{S} = \langle S, \models_S \rangle$ *be a denotational semantics for* $\mathcal{L}$*. Then* $\langle \mathcal{L}, \vdash_{\mathsf{S}} \rangle$ *is a propositional logic for* $\mathcal{L}$*.*

3 Inconsistency Tolerance

In the context of reasoning with uncertainty, a major drawback of a logic $\langle \mathcal{L}, \vdash_{\mathsf{S}} \rangle$, induced by a denotational semantics $\mathsf{S} = \langle S, \models_S \rangle$, is that it may not tolerate inconsistency properly. Indeed, if $mod_{\mathsf{S}}(\Gamma)$ is empty, then by Definition 4 it holds that $\Gamma \vdash_{\mathsf{S}} \psi$ for *every* formula ψ. We thus consider a 'refined' entailment relation, denoted $\mid\!\sim_{\mathsf{S}}$, that overcomes this explosive nature of $\vdash_{\mathsf{S}}$, but respects $\vdash_{\mathsf{S}}$ with respect to consistent theories. Formally, we require the following two properties:

I. Faithfulness: If $mod_{\mathsf{S}}(\Gamma) \neq \emptyset$ then for all $\psi \in \mathcal{F}_{\mathcal{L}}$, $\Gamma \mid\!\sim_{\mathsf{S}} \psi$ iff $\Gamma \vdash_{\mathsf{S}} \psi$.

II Non-Explosiveness: If $mod_{\mathsf{S}}(\Gamma) = \emptyset$ then there is a formula $\psi \in \mathcal{F}_{\mathcal{L}}$ such that $\Gamma \not\mid\!\sim_{\mathsf{S}} \psi$.

[1] Due to a lack of space, some proofs are omitted or shortened.

oarieli@mta.ac.il

Faithfulness guarantees that $\mid\!\sim_{\mathsf{S}}$ coincides with $\vdash_{\mathsf{S}}$ with respect to S-consistent theories, and non-explosiveness assures that $\mid\!\sim_{\mathsf{S}}$ is not trivialized when the set of premises is not S-consistent. We call $\mid\!\sim_{\mathsf{S}}$ an *inconsistency-tolerant* variant of $\vdash_{\mathsf{S}}$. In what follows, when $\vdash_{\mathsf{S}}$ is clear from context, we just say that $\mid\!\sim_{\mathsf{S}}$ is inconsistency-tolerant.

Note 1. When $mod_{\mathsf{S}}(\Gamma)$ is not empty for every Γ, then $\vdash_{\mathsf{S}}$ itself is inconsistency-tolerant, but in such logics the notion of inconsistency is degenerated. In what follows we shall be interested in stronger logics (like classical logic) that do not tolerate inconsistency and so need to be refined. Moreover, by Proposition 1, $\vdash_{\mathsf{S}}$ is a consequence relation and so it is monotonic, but frequently commonsense reasoning is nonmonotonic, in particular in the presence of contradictions.

4 Dissimilarity-Based Entailments

For defining inconsistency-tolerant variants of consequence relations that are induced by a denotational semantics, we introduce the notion of *dissimilarity*. Intuitively, dissimilarities are quantitative indications about the distinction between interpretations.

Definition 5. Let $\mathsf{S} = \langle S, \models_S \rangle$ be a denotational semantics. An S-*dissimilarity* is a function $\mathsf{d} : S \times S \to \mathbb{R}^+$, satisfying the following properties for all $\nu, \mu \in S$:

Symmetry: $\mathsf{d}(\nu, \mu) = \mathsf{d}(\mu, \nu)$,
Reflexivity: $\mathsf{d}(\nu, \nu) = 0$,
Absorption: if $\mathsf{d}(\nu, \mu) = 0$ then $\mathsf{d}(\nu, \sigma) = \mathsf{d}(\mu, \sigma)$ for every $\sigma \in S$.

Example 3. The discrete (uniform) metric d^{u} on S, defined by $\mathsf{d}^{\mathsf{u}}(\nu, \mu) = 0$ if $\nu = \mu$ and $\mathsf{d}^{\mathsf{u}}(\nu, \mu) = 1$ otherwise, is an S-dissimilarity. The Hamming distance d^{h} [11], where $\mathsf{d}^{\mathsf{h}}(\nu, \mu)$ is the number of atoms p for which $\nu(p) \neq \mu(p)$, is a dissimilarity on, e.g., two-valued valuations (Example 1) applied to a finite number of atomic formulas. Other definitions of distance and dissimilarity functions can be found, e.g., in [4, 5, 6, 17, 18].

Note 2. Dissimilarities are a generalization of the notion of pseudo distances: If d is a pseudo distance on S (that is, if d is a symmetric total function on S that preserves identities: $\forall \nu, \mu \in S$ $\mathsf{d}(\nu, \mu) = 0$ iff $\nu = \mu$), then d is also an S-dissimilarity. However, dissimilarities are a weaker notion than distances: a function d that satisfies all the conditions in Definition 5 is not necessarily a pseudo-distance, since $\mathsf{d}(\nu, \mu) = 0$ does not necessarily mean that ν and μ are equal (similarity does not preserve identities).

To be computable, dissimilarity functions should take into consideration only *finite* fragments of the compared interpretations (which are in general infinite). This is done by restricting the comparisons to finite contexts, determined by the underlying theory.

oarieli@mta.ac.il

Definition 6. Let $\mathsf{S} = \langle S, \models_S \rangle$ be a denotational semantics for $\mathcal{L}$. A *context* is a finite set of formulas (i.e., an element of $\mathcal{T}_{\mathcal{L}}$). A *context generator* (for $\mathcal{L}$) is a function $\mathcal{G} : \mathcal{T}_{\mathcal{L}} \to \mathcal{T}_{\mathcal{L}}$, producing a context for every theory.

Intuitively, $\mathcal{G}(\Gamma)$ is a relevant context for the computations about a theory Γ. In what follows we shall denote by $\mathcal{G} \subseteq \mathcal{G}'$ that $\mathcal{G}(\Gamma) \subseteq \mathcal{G}'(\Gamma)$ for every Γ.

Example 4. Common examples for context generators are, e.g., the functions $\mathcal{G}^{\mathsf{At}}$, $\mathcal{G}^{\mathsf{SF}}$, $\mathcal{G}^{\mathsf{ID}}$, respectively, defined for every theory Γ by $\mathcal{G}^{\mathsf{At}}(\Gamma) = \mathsf{Atoms}(\Gamma)$, $\mathcal{G}^{\mathsf{SF}}(\Gamma) = \mathsf{SF}(\Gamma)$, and $\mathcal{G}^{\mathsf{ID}}(\Gamma) = \Gamma$.

Dissimilarities will be used in what follows to determine how 'close' an interpretation is to satisfying a set of formulas Γ. In particular, they should differentiate between the models of Γ and non-models of Γ. However, as similarities are a weaker notion (and so more general) than pseudo-distances (Note 2), it may be the case that $\mathsf{d}(\nu, \mu) = 0$ where ν is a model of Γ while μ is not. To avoid this, the dissimilarity under consideration should be Γ-dependent. This is achieved by dissimilarity generators that generate different similarities for different theories. For defining dissimilarity generators we therefore need first the following notation and notion:

Definition 7. $\nu \sim_\Gamma \mu$ if for every $\psi \in \Gamma$, $\nu \models_S \psi$ iff $\mu \models_S \psi$.

Definition 8. We say that a formula ψ is $\mathcal{G}$*-independent* of a theory Γ, if $\mathcal{G}(\{\psi\}) \cap \mathcal{G}(\Gamma) = \emptyset$.

Definition 9. Let $\mathsf{S} = \langle S, \models_S \rangle$ be a denotational semantics for a language $\mathcal{L}$ and $\mathcal{G}$ a context generator for the same language. A $\mathcal{G}$*-dissimilarity generator for* S is a function $\mathsf{d}_{\mathcal{G}} : \mathcal{T}_{\mathcal{L}} \to (S \times S \to \mathbb{R}^+)$, such that:

1. For every $\Gamma \in \mathcal{T}_{\mathcal{L}}$, $\mathsf{d}_{\mathcal{G}}(\Gamma)$ is an S-dissimilarity.
2. For every $\nu, \mu \in S$, if $\mathsf{d}_{\mathcal{G}}(\Gamma)(\nu, \mu) = 0$, then $\nu \sim_\Gamma \mu$.

A $\mathcal{G}$-dissimilarity generator $\mathsf{d}_{\mathcal{G}}$ is called *normal*, if it satisfies the following normality condition:

– For every theory Γ, if ψ is a non S-tautological formula that is $\mathcal{G}$-independent of Γ, then for every $\nu \in mod_{\mathsf{S}}(\psi)$ there is $\mu \notin mod_{\mathsf{S}}(\psi)$ such that still $\mathsf{d}_{\mathcal{G}}(\Gamma)(\nu, \mu) = 0$.

Below, we shall sometimes write $\mathsf{d}_{\mathcal{G}(\Gamma)}$ instead of $\mathsf{d}_{\mathcal{G}}(\Gamma)$.

Note that $\mathsf{d}_{\mathcal{G}(\Gamma)}(\nu, \mu) = 0$ only means that ν and μ are similar on $\mathcal{G}(\Gamma)$, but this does not imply any correspondence between ν and μ elsewhere (cf. Note 2). Intuitively, the second condition of Definition 9 makes sure that $\mathsf{d}_{\mathcal{G}(\Gamma)}$ is faithful to Γ. The normality condition assures that the measurements by $\mathsf{d}_{\mathcal{G}(\Gamma)}$ depend only on the relevant context $\mathcal{G}(\Gamma)$.

Example 5. Let $\mathcal{G} = \mathcal{G}^{\mathsf{At}}$ or $\mathcal{G} = \mathcal{G}^{\mathsf{SF}}$, and let S be the standard two-valued semantics (Example 1). The following are all normal $\mathcal{G}$-dissimilarity generators for S.

oarieli@mta.ac.il

a) $\mathsf{d}^{\mathsf{s}}_{\mathcal{G}}$, where for every Γ, $\mathsf{d}^{\mathsf{s}}_{\mathcal{G}(\Gamma)}(\nu,\mu)=0$ if $\nu \sim_\Gamma \mu$, otherwise $\mathsf{d}^{\mathsf{s}}_{\mathcal{G}(\Gamma)}(\nu,\mu)=1$.
b) $\mathsf{d}^{\mathsf{u}}_{\mathcal{G}}$, where for every Γ, $\mathsf{d}^{\mathsf{u}}_{\mathcal{G}(\Gamma)}(\nu,\mu)=0$ if for all $\psi \in \mathcal{G}(\Gamma)$ $\nu(\psi)=\mu(\psi)$ and otherwise $\mathsf{d}^{\mathsf{u}}_{\mathcal{G}(\Gamma)}(\nu,\mu)=1$.
c) $\mathsf{d}^{\mathsf{h}}_{\mathcal{G}}$, in which $\mathsf{d}^{\mathsf{h}}_{\mathcal{G}(\Gamma)}(\nu,\mu)$ is the number of formulas $\psi \in \mathcal{G}(\Gamma)$, for which $\nu(\psi) \neq \mu(\psi)$.

In Section 6 we consider other (normal) dissimilarity generators. Note that unlike related formalisms in standard two-values semantics (e.g., [18, 21]), it is *not necessary* to assume here that (the set of atomic formulas in) the underlying language is finite. This is because the dissimilarity calculations are made with respect to finite contexts.

Definition 10. A *(numeric) aggregation function* is a total function f, such that: (1) for every multiset of real numbers, the value of f is a real number, (2) the value of f does not decrease when the number in its multiset increases, (3) $f(\{x_1,\ldots,x_n\})=0$ iff $x_1=x_2=\ldots x_n=0$, and (4) $\forall x \in \mathbb{R}$ $f(\{x\})=x$.

For non-negative values (such as dissimilarities), summation, average, median, and the maximum function, are all aggregation functions (see, e.g., [18, 23]).

Definition 11. Let S be a denotational semantics for a language $\mathcal{L}$. A (semantical) *setting* for S is a triple $\mathcal{S}=\langle \mathcal{G},\mathsf{d},f\rangle$, where $\mathcal{G}$ is a context generator (for $\mathcal{L}$), d is a $\mathcal{G}$-dissimilarity generator (for S), and f is a numeric aggregation function.

Definition 12. Let $\mathcal{S}=\langle \mathcal{G},\mathsf{d},f\rangle$ be a setting for a denotational semantics S. For a theory $\Gamma=\{\psi_1,\ldots,\psi_n\}$, define:

$$\mathsf{m}^{\Gamma}_{\mathcal{S}}(\nu,\psi_i)=\begin{cases} min\big\{\mathsf{d}_{\mathcal{G}(\Gamma)}(\nu,\mu) \mid \mu \in mod_{\mathsf{S}}(\psi_i)\big\} & \text{if } mod_{\mathsf{S}}(\psi_i)\neq\emptyset,\\ 1+max\big\{\mathsf{d}_{\mathcal{G}(\Gamma)}(\nu,\mu) \mid \nu,\mu \in S\big\} & \text{otherwise.}\end{cases}$$

$$\mathsf{M}_{\mathcal{S}}(\nu,\Gamma)=f\big(\big\{\mathsf{m}^{\Gamma}_{\mathcal{S}}(\nu,\psi_1),\ldots,\mathsf{m}^{\Gamma}_{\mathcal{S}}(\nu,\psi_n)\big\}\big).$$

Intuitively, $\mathsf{m}^{\Gamma}_{\mathcal{S}}(\nu,\psi)$ is a quantitative indication for how 'close' ν is to be a model of ψ. The function $\mathsf{M}_{\mathcal{S}}(\nu,\Gamma)$ indicates how 'close' ν is to satisfy Γ. It is easy to verify that if ψ is S-consistent, then the closest elements to ψ are its models, and if ψ is *not* S-consistent, all the elements in S are equally close to ψ.

Definition 13. A setting $\mathcal{S}=\langle \mathcal{G},\mathsf{d},f\rangle$ for a denotational semantics $\mathsf{S}=\langle S,\models_S\rangle$ is *normal*, if so is d (recall Definition 9). We say that $\mathcal{S}$ is *effective*, if for every theory Γ, the set $\{\mathsf{M}_{\mathcal{S}}(\nu,\Gamma) \mid \nu \in S\}$ has a minimal element.

Reasoning by similarities is now defined as follows:

Definition 14. Given a setting $\mathcal{S}$ for a denotational semantics $\mathsf{S}=\langle S,\models_S\rangle$, the *$\mathcal{S}$-most plausible interpretations* of a (nonempty) theory Γ, are the elements of the following set:

$$\Delta_{\mathcal{S}}(\Gamma)=\big\{\nu \in S \mid \forall \mu \in S\ \mathsf{M}_{\mathcal{S}}(\nu,\Gamma)\leq \mathsf{M}_{\mathcal{S}}(\mu,\Gamma)\big\}.$$

In case that Γ is empty, we define $\Delta_{\mathcal{S}}(\emptyset)=S$.

oarieli@mta.ac.il

Note 3. If $\mathcal{S}$ is effective, then $\Delta_{\mathcal{S}}(\Gamma) \neq \emptyset$ for every Γ.

Definition 15. Given a semantic setting $\mathcal{S}$ for a denotational semantics S, define: $\Gamma \mid\!\sim_{\mathcal{S}} \psi$ iff $\Delta_{\mathcal{S}}(\Gamma) \subseteq mod_{\mathsf{S}}(\psi)$.

5 Reasoning with $\mid\!\sim_{\mathcal{S}}$

The following example is a simple illustration of how inconsistency is handled by a dissimilarity-based entailment.

Example 6. Consider the setting $\mathcal{S} = \langle \mathcal{G}^{\mathsf{At}}, \mathsf{d}^{\mathsf{h}}, \Sigma \rangle$ for the standard two-valued semantics (Example 1), where $\mathsf{d}^{\mathsf{h}} = \mathsf{d}^{\mathsf{h}}_{\mathcal{G}^{\mathsf{At}}}$ is a dissimilarity generator in Example 5c and Σ is the summation function. Now, let $\Gamma = \{p, \neg p, q\}$. In this case, $\Delta_{\mathcal{S}}(\Gamma) = \{\nu \in S \mid \nu(q) = t\}$, thus $\Gamma \mid\!\sim_{\mathcal{S}} q$ while $\Gamma \not\mid\!\sim_{\mathcal{S}} \neg q$. This is justified by the fact that q has nothing to do with the inconsistency of Γ (more precisely, q is not $\mathcal{G}^{\mathsf{At}}$-dependent of $\{p, \neg p\}$). On the other hand, $\Gamma \not\mid\!\sim_{\mathcal{S}} p$ and $\Gamma \not\mid\!\sim_{\mathcal{S}} \neg p$, since both of p and $\neg p$ are 'involved' in the inconsistency of Γ.

Next, we consider some of the main properties of $\mid\!\sim_{\mathcal{S}}$.

Theorem 1. *Let $\mathcal{S} = \langle \mathcal{G}, \mathsf{d}, f \rangle$ be an effective setting for a denotational semantics S. If S is normal (Definition 3), or $\mathcal{S}$ is normal (Definition 13), then $\mid\!\sim_{\mathcal{S}}$ is an inconsistency-tolerant variant of $\vdash_{\mathsf{S}}$.*

Proof outline. It is easy to verify that for every formula ψ, $\mathsf{m}^{\Gamma}_{\mathcal{S}}(\nu, \psi) = 0$ iff $\nu \in mod_{\mathsf{S}}(\psi)$. Thus, for every theory Γ, $\mathsf{M}_{\mathcal{S}}(\nu, \Gamma) = 0$ iff $\nu \in mod_{\mathsf{S}}(\Gamma)$. It follows that if Γ is S-consistent, $\Delta_{\mathcal{S}}(\Gamma) = mod_{\mathsf{S}}(\Gamma)$, and so $\Gamma \mid\!\sim_{\mathcal{S}} \psi$ iff $\Gamma \vdash_{\mathsf{S}} \psi$. This shows faithfulness. Now, if S is normal, then non-explosiveness follows from the fact that as $\mathcal{S}$ is effective, $\Delta_{\mathcal{S}}(\Gamma)$ is nonempty (Note 3), thus for every $\nu \in \Delta_{\mathcal{S}}(\Gamma)$ there is some ψ such that $\nu \not\models_S \psi$, and so $\Gamma \not\mid\!\sim_{\mathsf{S}} \psi$. If $\mathcal{S}$ is normal, non-explosiveness follows from the following lemma:

Lemma 1. *Let $\mathcal{S} = \langle \mathcal{G}, \mathsf{d}, f \rangle$ be an effective and normal setting for S. For every Γ and every ψ which is $\mathcal{G}$-independent of Γ, it holds that $\Gamma \mid\!\sim_{\mathcal{S}} \psi$ iff ψ is an S-tautology.*

Proof. One direction is clear: if ψ is an S-tautology, then for every $\nu \in \Delta_{\mathcal{S}}(\Gamma)$, $\nu \models_S \psi$ and so $\Gamma \mid\!\sim_{\mathcal{S}} \psi$. For the converse, suppose that ψ is not an S-tautology. Since $\mathcal{S}$ is effective, by Note 3 there is a valuation $\nu \in \Delta_{\mathcal{S}}(\Gamma)$. If $\nu \not\models_S \psi$, we are done: $\Gamma \not\mid\!\sim_{\mathcal{S}} \psi$. Otherwise, since $\mathsf{d}_{\mathcal{G}}$ is normal, there is some $\mu \in S$ such that $\mu \not\models_S \psi$, and $\mathsf{d}_{\mathcal{G}(\Gamma)}(\nu, \mu) = 0$. But $\mathsf{d}_{\mathcal{G}(\Gamma)}$ is an S-dissimilarity and so, by Absorption (Definition 5), for every $\psi \in \Gamma$ and every $\nu_0 \in mod_S(\psi)$, $\mathsf{d}_{\mathcal{G}(\Gamma)}(\nu, \nu_0) = \mathsf{d}_{\mathcal{G}(\Gamma)}(\mu, \nu_0)$. Hence $\mathsf{m}^{\Gamma}_{\mathcal{S}}(\nu, \psi) = \mathsf{m}^{\Gamma}_{\mathcal{S}}(\mu, \psi)$, and so $\mathsf{M}_{\mathcal{S}}(\nu, \Gamma) = \mathsf{M}_{\mathcal{S}}(\mu, \Gamma)$ as well. Now, since $\nu \in \Delta_{\mathcal{S}}(\Gamma)$, it holds also that $\mu \in \Delta_{\mathcal{S}}(\Gamma)$, and since $\mu \not\models_S \psi$, we have that $\Gamma \not\mid\!\sim_{\mathcal{S}} \psi$. □

oarieli@mta.ac.il

Theorem 2. *Let $\mathcal{S} = \langle \mathcal{G}, \mathsf{d}, f \rangle$ be an effective setting for a denotational semantics S, in which f is hereditary*[2]*. Then $\mid\!\sim_{\mathcal{S}}$ is a cautious consequence relation (in the sense of [2]), i.e., it has the following properties:*

Cautious Reflexivity: if Γ is S-satisfiable and $\psi \in \Gamma$ then $\Gamma \mid\!\sim_{\mathcal{S}} \psi$

Cautious Monotonicity [14]: if $\Gamma \mid\!\sim_{\mathcal{S}} \psi$ and $\Gamma \mid\!\sim_{\mathcal{S}} \phi$ then $\Gamma, \psi \mid\!\sim_{\mathcal{S}} \phi$

Cautious Transitivity [19]: if $\Gamma \mid\!\sim_{\mathcal{S}} \psi$ and $\Gamma, \psi \mid\!\sim_{\mathcal{S}} \phi$ then $\Gamma \mid\!\sim_{\mathcal{S}} \phi$

The next proposition shows that many entailments of the form $\mid\!\sim_{\mathcal{S}}$ are *paraconsistent* [10].

Proposition 2. *Let S be a denotational semantics for a language $\mathcal{L}$ with a unary connective $\neg$. Let $\mathcal{S} = \langle \mathcal{G}, \mathsf{d}, f \rangle$ be an effective setting for S, in which $\mathcal{G} \subseteq \mathcal{G}^{\mathsf{SF}}$. Then $\mid\!\sim_{\mathcal{S}}$ is $\neg$-paraconsistent: $\psi, \neg\psi \not\mid\!\sim_{\mathcal{S}} \phi$ for some formulas $\psi, \phi \in \mathcal{F}_{\mathcal{L}}$.*

Another interesting property of dissimilarity-based entailments is that their set of conclusions is always consistent.

Proposition 3. *Let $\mathcal{S} = \langle \mathcal{G}, \mathsf{d}, f \rangle$ be an effective setting for a denotational semantics S. Then for every theory Γ the set of the formulas that are $\mid\!\sim_{\mathcal{S}}$-entailed by Γ is S-consistent.*

Proof. Let $\mathcal{C}_{\mathcal{S}}(\Gamma) = \{\psi \mid \Gamma \mid\!\sim_{\mathcal{S}} \psi\}$. If $\mathcal{C}_{\mathcal{S}}(\Gamma)$ is not S-consistent for some Γ, that is, if $mod_{\mathsf{S}}(\mathcal{C}_{\mathcal{S}}(\Gamma)) = \emptyset$, then since $\Delta_{\mathcal{S}}(\Gamma) \subseteq mod_{\mathsf{S}}(\psi)$ for every $\psi \in \mathcal{C}_{\mathcal{S}}(\Gamma)$, we have: $\Delta_{\mathcal{S}}(\Gamma) \subseteq \bigcap_{\psi \in \mathcal{C}_{\mathcal{S}}(\Gamma)} mod_{\mathsf{S}}(\psi) = mod_{\mathsf{S}}(\mathcal{C}_{\mathcal{S}}(\Gamma)) = \emptyset$. Thus $\Delta_{\mathcal{S}}(\Gamma) = \emptyset$, contradicting the fact that if $\mathcal{S}$ is effective, $\Delta_{\mathcal{S}}(\Gamma)$ is nonempty for every Γ (see Note 3). □

Finally, we consider the decidability of $\mid\!\sim_{\mathcal{S}}$.

Definition 16. A setting $\mathcal{S} = \langle \mathcal{G}, \mathsf{d}_{\mathcal{G}}, f \rangle$ is *computable*, if the following conditions hold:

1. The functions $\mathcal{G}$, f, $\mathsf{d}_{\mathcal{G}}$ and $\mathsf{d}_{\mathcal{G}(\Gamma)}$ for every theory Γ, are all computable.
2. $\mathsf{d}_{\mathcal{G}(\Gamma)}$ is bounded: for every Γ there is $n_\Gamma \in \mathbb{N}$ such that $\mathsf{d}_{\mathcal{G}(\Gamma)}(\nu, \mu) \leq n_\Gamma$ for all $\nu, \mu \in S$.
3. For every theory Γ, formula ψ, and number $i \in \mathbb{N}$, checking whether $\{\nu \in S \mid \mathsf{M}_{\mathcal{S}}(\nu, \Gamma) = i\} \subseteq mod_{\mathsf{S}}(\psi)$ is decidable.

Example 7. The setting $\mathcal{S} = \langle \mathcal{G}, \mathsf{d}_{\mathcal{G}}, f \rangle$ for standard two-valued semantics is computable for every context generator $\mathcal{G}$ in Example 4, all the similarity generators $\mathsf{d}_{\mathcal{G}}$ in Example 5, and every computable aggregation function f.

Theorem 3. *If $\mathcal{S}$ is computable then $\Gamma \mid\!\sim_{\mathcal{S}} \psi$ is decidable.*

6 Applications

We now demonstrate the usefulness (and generality) of dissimilarity reasoning for defining a variety of inconsistency-tolerant logics based on different types of denotational semantics.

[2] An aggregation function f is called hereditary, if $f(\{x_1, \ldots, x_n\}) < f(\{y_1, \ldots, y_n\})$ implies that $f(\{x_1, \ldots, x_n, z_1, \ldots, z_m\}) < f(\{y_1, \ldots, y_n, z_1, \ldots, z_m\})$ (see [1]).

oarieli@mta.ac.il

6.1 Multi-valued Logics

The most standard way of defining multi-valued logics (including, of course, classical logic), is by the following structures (see, e.g., [25]):

Definition 17. A *(multi-valued) matrix* for a language $\mathcal{L}$ is a triple $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$, where $\mathcal{V}$ is a non-empty set of truth values, $\mathcal{D}$ is a non-empty proper subset of $\mathcal{V}$, and $\mathcal{O}$ contains an interpretation $\tilde{\diamond} : \mathcal{V}^n \to \mathcal{V}$ for each n-ary connective of $\mathcal{L}$.

Given a matrix $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$, we shall assume that $\mathcal{V}$ includes at least the two classical values t and f, and that only the former belongs to the set $\mathcal{D}$ of the *designated elements* in $\mathcal{V}$ (representing 'true assertions'). The set $\mathcal{O}$ contains the interpretations (the 'truth tables') of each connective in $\mathcal{L}$. The associated semantical notions are now defined as usual: An $\mathcal{M}$*-valuation* is a function $\nu : \mathcal{F}_{\mathcal{L}} \to \mathcal{V}$ so that, for every connective $\diamond$ in $\mathcal{L}$, $\nu(\diamond(\psi_1, \ldots, \psi_n)) = \tilde{\diamond}(\nu(\psi_1), \ldots, \nu(\psi_n))$. The set of all $\mathcal{M}$-valuations is denoted by $\Lambda_{\mathcal{M}}$. We say that ν is a *model* of ψ, denoted $\nu \models_{\mathcal{M}} \psi$, if $\nu(\psi) \in \mathcal{D}$.

Note that the pair $\langle \Lambda_{\mathcal{M}}, \models_{\mathcal{M}} \rangle$ is a denotational semantics in the sense of Definition 2. In what follows we shall sometimes identify this semantics with the matrix $\mathcal{M}$ that defines it. In particular, we shall say that $\mathcal{M}$ is normal if so is the denotational semantics $\langle \Lambda_{\mathcal{M}}, \models_{\mathcal{M}} \rangle$ that it induces. By Proposition 1 we have, then, that:

Proposition 4. *The relation $\vdash_{\mathcal{M}}$, induced from a matrix $\mathcal{M}$ by Definition 4, is a Tarskian consequence relation.*

Given a matrix-based denotational semantics $\langle \Lambda_{\mathcal{M}}, \models_{\mathcal{M}} \rangle$ and a corresponding consequence relation $\vdash_{\mathcal{M}}$, one may define inconsistency-tolerant variants of $\vdash_{\mathcal{M}}$ by a dissimilarity-based setting $\mathcal{S} = \langle \mathcal{G}, \mathsf{d}, f \rangle$, just as in Definition 15.

Example 8. Kleene's three-valued logic [16] is obtained from the matrix $\mathcal{M}_K^3 = \langle \{t, f, \bot\}, \{t\}, \mathcal{O}_K \rangle$, in which $\mathcal{O}_K$ contains the following connective interpretations:

	$\tilde{\neg}$
f	t
$\bot$	$\bot$
t	f

$\tilde{\wedge}$	f	$\bot$	t
f	f	f	f
$\bot$	f	$\bot$	$\bot$
t	f	$\bot$	t

$\tilde{\vee}$	f	$\bot$	t
f	f	$\bot$	t
$\bot$	$\bot$	$\bot$	t
t	t	t	t

Let $\mathcal{S} = \langle \mathcal{G}^{\mathsf{At}}, \mathsf{d}, \Sigma \rangle$ be a setting, where $\mathcal{G}^{\mathsf{At}}$ is the atom-based context generator, Σ is a summation function, and d is a $\mathcal{G}^{\mathsf{At}}$-dissimilarity generator, producing for each Γ the following dissimilarity:

$$\mathsf{d}(\Gamma)(\nu, \mu) = \Sigma \, \{d_{\Sigma}(\nu(p), \mu(p)) \mid p \in \mathcal{G}^{\mathsf{At}}(\Gamma)\}.$$

Here, d_{Σ} is an extended Hamming distance on $\{t, f, \bot\}$, where $d_{\Sigma}(t, f) = 1$ and $d_{\Sigma}(t, \bot) = d_{\Sigma}(f, \bot) = 0.5$ (see [1, 12]). Let $\Gamma = \{\neg p, \neg q, p \vee q\}$. Clearly, Γ is

oarieli@mta.ac.il

not $\mathcal{M}_K^3$-satisfiable. We compute its most plausible interpretations with respect to $\mathcal{S}$:

	p	q	$\neg p$	$\neg q$	$p \vee q$	1	2	3	$\mathsf{M}_{\mathcal{S}}(\nu_i, \Gamma)$
ν_1	t	t	f	f	t	1	1	0	2
ν_2	t	f	f	t	t	1	0	0	**1**
ν_3	t	$\perp$	f	$\perp$	t	1	0.5	0	1.5
ν_4	f	t	t	f	t	0	1	0	**1**
ν_5	f	f	t	t	f	0	0	1	**1**
ν_6	f	$\perp$	t	$\perp$	$\perp$	0	0.5	0.5	**1**
ν_7	$\perp$	t	$\perp$	f	t	0.5	1	0	1.5
ν_8	$\perp$	f	$\perp$	t	$\perp$	0.5	0	0.5	**1**
ν_9	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	0.5	0.5	0.5	1.5

Legend. $1 = \mathsf{m}_{\mathcal{S}}^{\Gamma}(\nu_i, \neg p)$, $2 = \mathsf{m}_{\mathcal{S}}^{\Gamma}(\nu_i, \neg q)$, $3 = \mathsf{m}_{\mathcal{S}}^{\Gamma}(\nu_i, p \vee q)$.

Hence, $\Delta_{\mathcal{S}}(\Gamma) = \{\nu_2, \nu_4, \nu_5, \nu_6, \nu_8\}$, and so, for instance, $\Gamma \mid\!\sim_{\mathcal{S}} \neg p \vee \neg q$ (even though $\Gamma \not\mid\!\sim_{\mathcal{S}} \neg p$ and $\Gamma \not\mid\!\sim_{\mathcal{S}} \neg q$).

Let us now describe a general method of defining dissimilarity generators in multi-valued semantics.

Proposition 5. *Let $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$ be a matrix for a language $\mathcal{L}$, d a pseudo-distance on $\mathcal{V}$, g an aggregation function, and $\mathcal{G}$ a context generator for $\mathcal{L}$. We define, for every $\Gamma \in \mathcal{T}_{\mathcal{L}}$, a function $\mathsf{d}^g_{\mathcal{G}(\Gamma)} : \Lambda_{\mathcal{M}} \times \Lambda_{\mathcal{M}} \to \mathbb{R}^+$ by:*

$$\mathsf{d}^g_{\mathcal{G}(\Gamma)}(\nu, \mu) = g\big(\{d(\nu(\psi), \mu(\psi)) \mid \psi \in \mathcal{G}(\Gamma)\}\big).$$

Then $\mathsf{d}^g_{\mathcal{G}((\Gamma)}$ is a dissimilarity function. Moreover,

- *if $\mathcal{G}^{\mathsf{At}} \subseteq \mathcal{G}$ or $\mathcal{G}^{\mathsf{ID}} \subseteq \mathcal{G}$, $\mathsf{d}^g_{\mathcal{G}}$ is a $\mathcal{G}$-dissimilarity generator,*
- *if $\mathcal{G}^{\mathsf{At}} \subseteq \mathcal{G}$ and $\mathcal{G}^{\mathsf{At}} \circ \mathcal{G} \subseteq \mathcal{G}$,[3] then $\mathsf{d}^g_{\mathcal{G}}$ is a normal $\mathcal{G}$-dissimilarity generator.*

Given a propositional logic $\langle \mathcal{L}, \vdash_{\mathcal{M}} \rangle$, Proposition 5 yields a simple and a general way of defining entailments that are inconsistency-tolerant variants of $\vdash_{\mathcal{M}}$. We note that many of the inconsistency-tolerant entailments that have been considered in the literature (e.g., those in [15] and [18]) are particular cases of this construction, where $\mathcal{M}$ is a two-valued matrix and $\mathcal{G} = \mathcal{G}^{\mathsf{At}}$.

6.2 Non-deterministic Logics

Matrix-based semantics is truth-functional in the sense that the truth-value of a complex formula is determined by the truth-values of its subformulas. Such a semantics is not useful in capturing non-deterministic phenomena. This leads to the idea of non-deterministic matrices [7], allowing non-deterministic evaluations of formulas. This kind of semantics has a variety of applications for reasoning under uncertainty (see, e.g., [8]).

[3] I.e., $\mathcal{G}^{\mathsf{At}}(\mathcal{G}(\Gamma)) \subseteq \mathcal{G}(\Gamma)$ for every Γ.

oarieli@mta.ac.il

Definition 18. A *non-deterministic matrix* (*Nmatrix*) for $\mathcal{L}$ is a tuple $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$, where $\mathcal{V}$ is a non-empty set of truth values, $\mathcal{D}$ is a non-empty proper subset of $\mathcal{V}$, and $\mathcal{O}$ contains an interpretation function $\tilde{\diamond} : \mathcal{V}^n \to 2^{\mathcal{V}} \setminus \{\emptyset\}$ for every n-ary connective of $\mathcal{L}$.

An $\mathcal{M}$-*valuation* is a function $\nu : \mathcal{F}_{\mathcal{L}} \to \mathcal{V}$ such that for every connective $\diamond$ in $\mathcal{L}$,

$$\nu(\diamond(\psi_1, \ldots, \psi_n)) \in \tilde{\diamond}(\nu(\psi_1), \ldots, \nu(\psi_n)).$$

The set of all $\mathcal{M}$-valuations is denoted by $\Lambda_{\mathcal{M}}$. Again, $\nu \in \Lambda_{\mathcal{M}}$ is an $\mathcal{M}$-*model* of ψ (denoted $\nu \models_{\mathcal{M}} \psi$), if $\nu(\psi) \in \mathcal{D}$.

Ordinary matrices can be thought of as Nmatrices whose interpretations return singletons of truth-values. Again, for an Nmatrix $\mathcal{M}$, the pair $\langle \Lambda_{\mathcal{M}}, \models_{\mathcal{M}} \rangle$ is a denotational semantics and it induces a Tarskian consequence relation $\vdash_{\mathcal{M}}$.

As before, dissimilarity-based entailments can be applied for non-deterministic matrices. However, as was shown in [5], some dissimilarity generators (and the respective settings) that are definable with respect to standard matrices, are *not* applicable in the non-deterministic case. This is due to the fact that non-deterministic valuations are not truth functional, so they can agree on atomic formulas, but may make different non-deterministic choices on complex formulas. This is also the reason why Proposition 5 is not extendable to non-deterministic semantics. Yet, a stricter version of that proposition does hold also in the non-deterministic case:

Proposition 6. *Let $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$ be an Nmatrix for a language $\mathcal{L}$, d a pseudo-distance on $\mathcal{V}$, g an aggregation function, and $\mathcal{G}$ a context generator for $\mathcal{L}$. We define, for each $\Gamma \in \mathcal{T}_{\mathcal{L}}$, a function $\mathsf{d}^g_{\mathcal{G}(\Gamma)} : \Lambda_{\mathcal{M}} \times \Lambda_{\mathcal{M}} \to \mathbb{R}^+$ by:*

$$\mathsf{d}^g_{\mathcal{G}(\Gamma)}(\nu, \mu) = g\big(\{d(\nu(\psi), \mu(\psi)) \mid \psi \in \mathcal{G}(\Gamma)\}\big).$$

Then $\mathsf{d}^g_{\mathcal{G}((\Gamma)}$ is a dissimilarity function. Moreover,

- *if $\mathcal{G}^{\mathsf{ID}} \subseteq \mathcal{G}$ then $\mathsf{d}^g_{\mathcal{G}}$ is a $\mathcal{G}$-dissimilarity generator,*
- *if $\mathcal{G}^{\mathsf{SF}} \subseteq \mathcal{G}$ and $\mathcal{G}^{\mathsf{SF}} \circ \mathcal{G} \subseteq \mathcal{G}$, then $\mathsf{d}^g_{\mathcal{G}}$ is a normal $\mathcal{G}$-dissimilarity generator.*

Note 4. Clearly, there are useful dissimilarity generators other than those that are covered by the construction of Proposition 6. One of them is the dissimilarity generator $\mathsf{d}^{\mathsf{n}}_{\mathcal{G}}$, in which $\mathsf{d}^{\mathsf{n}}_{\mathcal{G}(\Gamma)}(\nu, \mu)$ is the number of formulas $\psi \in \mathcal{G}(\Gamma)$, for which (i) $\nu(\psi) \neq \mu(\psi)$, and (ii) if $\psi = \diamond(\varphi_1, \ldots, \varphi_n)$, then for all $1 \leq i \leq n$: $\nu(\varphi_i) = \mu(\varphi_i)$. This function is one of the pseudo-distances that are introduced in [5] for distance-based reasoning for non-deterministic matrices. It can be verified that $\mathsf{d}^{\mathsf{n}}_{\mathcal{G}}$ is also a $\mathcal{G}$-dissimilarity generator in our sense.

Example 9. Consider a transmission protocol for a system with three transmitters T_1, T_2 and T_3, where the first two are connected to a bus through an arbiter A, and the third one is connected directly to the bus. The bus has a line Msg for the transmitted message, and a line Busy, which is turned on whenever the

oarieli@mta.ac.il

transmission occurs. When one of the transmitters T_1 or T_2 has a message to transmit, it signals to the arbiter by turning on the line M_1 or M_2 respectively. The arbiter then turns on the line Busy, and T_i transmits its message on the line Msg. As for the third transmitter, whenever T_3 wants to transmit a message, it turns on M_3 and transmits the message on Msg. A schematic presentation of this circuit (excluding some issues about the logic of Msg, which are not relevant for this example) is shown in Figure 1.

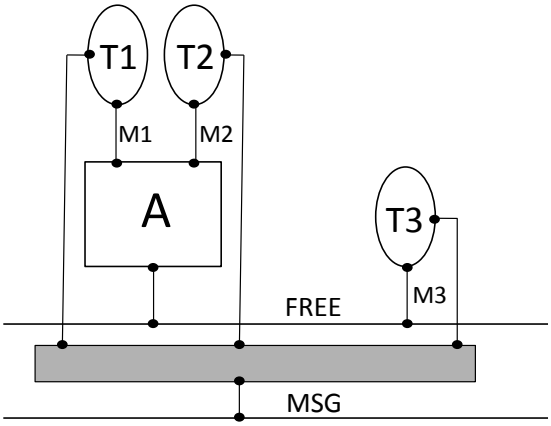


Fig. 1. The system of Example 9

Suppose now that the arbiter has no synchronization method, and whenever T_1 and T_2 request the line at the same time, the result is unpredictable: the line Busy can either stay on or be turned off. This non-deterministic behavior of the arbiter can be described using the following interpretation of the connective $\odot$

$\tilde{\odot}$	f	t
f	$\{f\}$	$\{t\}$
t	$\{t\}$	$\{t, f\}$

Let $\mathcal{M}$ be the two-valued Nmatrix for the language of $\{\neg, \vee, \odot\}$, which includes the above non-deterministic interpretation for $\odot$ and the standard interpretations of negation $\neg$ and disjunction $\vee$.

Next, suppose that we observe the following unexpected behavior of the arbiter: although T_1 has a message to transmit, while T_2 has none, the line Busy is not turned on. This can be captured by the following theory:

$$\Gamma = \{M_1, \neg M_2, \neg \mathsf{Busy}\}$$

where Busy is an abbreviation of the formula $(M_1 \odot M_2) \vee M_3$, representing the normal behavior of the line Busy. Obviously, this theory is not $\mathcal{M}$-satisfiable. For reasoning with this abnormality, we use the setting $\mathcal{S} = \langle \mathcal{G}^{\mathsf{SF}}, \mathsf{d}^{\mathsf{n}}_{\mathcal{G}^{\mathsf{SF}}}, \Sigma \rangle$, where $\mathsf{d}^{\mathsf{n}}_{\mathcal{G}^{\mathsf{SF}}}$

oarieli@mta.ac.il

is the dissimilarity generator defined in Note 4. The dissimilarity computations for this case are represented in the table below:

	M_1	M_2	M_3	$\neg M_2$	$M_1 \odot M_2$	Busy	$\neg$Busy	1	2	3	$\mathsf{M}_{\mathcal{S}}(\nu_i, \Gamma)$
ν_1	t	t	t	f	t	t	f	0	1	2	3
ν_2	t	t	t	f	f	t	f	0	1	1	2
ν_3	t	t	f	f	t	t	f	0	1	1	2
ν_4	t	t	f	f	f	f	t	0	1	0	**1**
ν_5	t	f	t	t	t	t	f	0	0	2	2
ν_6	t	f	f	t	t	t	f	0	0	1	**1**
ν_7	f	t	t	f	t	t	f	1	1	2	4
ν_8	f	t	f	f	t	t	f	1	1	1	3
ν_9	f	f	t	t	f	t	f	1	0	1	2
ν_{10}	f	f	f	t	f	f	t	1	0	0	**1**

Legend. $1 = \mathsf{m}^{\Gamma}_{\mathcal{S}}(\nu_i, M_1)$, $2 = \mathsf{m}^{\Gamma}_{\mathcal{S}}(\nu_i, \neg M_2)$, $3 = \mathsf{m}^{\Gamma}_{\mathcal{S}}(\nu_i, \neg\mathsf{Busy})$.

Hence, $\Delta_{\mathcal{S}}(\Gamma) = \{\nu_4, \nu_6, \nu_{10}\}$, and so, for instance, $\Gamma \mid\!\sim_{\mathcal{S}} \neg M_3$, even though neither M_1 nor $\neg M_2$ are $\mid\!\sim_{\mathcal{S}}$-inferrable from Γ. Hence, although one can say nothing about T_1 and T_2, it is still possible to conclude in this case that T_3 has no message to transmit.

6.3 Modal Logics

Next, we consider a denotational semantics that is based on a many-valued extension of standard Kripke semantics (see [13]), where the logical connectives are interpreted by a matrix $\mathcal{M}$,[4] and qualifications of the truth of a judgement are expressed by the necessitation operator "$\Box$". In case of the standard two-valued matrix we get the usual Kripke-style (possible worlds) semantics.

Definition 19. Let $\mathcal{L}$ be a propositional language.

- A *frame* for $\mathcal{L}$ is a triple $\mathcal{F} = \langle W, R, \mathcal{M}\rangle$, where W is a non-empty set (of "worlds"), R (the "accessibility relation") is a binary relation on W, and $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O}\rangle$ is a matrix for $\mathcal{L}$. We say that a frame is finite if so is W.
- Let $\mathcal{F} = \langle W, R, \mathcal{M}\rangle$ be a frame for $\mathcal{L}$. An $\mathcal{F}$*-valuation* is a function $\nu : W \times F_{\mathcal{L}} \to \mathcal{V}$ that assigns truth values to the $\mathcal{L}$-formulas at each world in W according to the following conditions:
  - For every connective $\diamond$, $\nu(w, \diamond(\psi_1, \ldots, \psi_n)) = \tilde{\diamond}_{\mathcal{M}}(\nu(w, \psi_1), \ldots, \nu(w, \psi_n))$,
  - $\nu(w, \Box\psi) \in \mathcal{D}$ iff $\nu(w', \psi) \in \mathcal{D}$ for all w' such that $R(w, w')$.

  The set of $\mathcal{F}$-valuations is denoted by $\Lambda_{\mathcal{F}}$. The set of $\mathcal{F}$-valuations that satisfy a formula ψ in a world $w \in W$ is $mod^{w}_{\mathcal{F}}(\psi) = \{\nu \in \Lambda_{\mathcal{F}} \mid \nu(w, \psi) \in \mathcal{D}\}$.
- A *frame interpretation* is a pair $I = \langle \mathcal{F}, \nu\rangle$, in which $\mathcal{F} = \langle W, R, \mathcal{M}\rangle$ is a frame and ν is an $\mathcal{F}$-valuation. We say that I *satisfies* ψ (or that I is a model of ψ), if $\nu \in mod^{w}_{\mathcal{F}}(\psi)$ for every $w \in W$. We say that I satisfies Γ if it satisfies every $\psi \in \Gamma$.

[4] This framework can be extended to Nmatrices as well, but for simplicity we stick to deterministic matrices.

oarieli@mta.ac.il

Let $\mathcal{I}$ be a nonempty set of frame interpretations. We define a satisfaction relation $\models_{\mathcal{I}}$ on $\mathcal{I} \times F_{\mathcal{L}}$ by $I \models_{\mathcal{I}} \psi$ iff I satisfies ψ. Note that $\mathfrak{I} = \langle \mathcal{I}, \models_{\mathcal{I}} \rangle$ is a denotational semantics in the sense of Definition 2. By Proposition 1, then, the induced relation $\vdash_{\mathfrak{I}}$ is a Tarskian consequence relation for $\mathcal{L}$.

Given a possible-world semantics $\mathfrak{I} = \langle \mathcal{I}, \models_{\mathcal{I}} \rangle$, it is possible to define an inconsistency-tolerant variant of $\vdash_{\mathfrak{I}}$ by introducing a dissimilarity-based setting $\mathcal{S} = \langle \mathcal{G}, \mathsf{d}, f \rangle$ and applying the definitions in Section 4. As frame interpretations are more complicated semantic structures than those considered in the previous sections, defining intuitive and simple dissimilarity generators is more challenging in this case. Below, we consider a simple and useful case: a set of frame interpretations in which all the frames share the same set of worlds and accessibility relation. In this case, dissimilarity between frame interpretations may be defined by comparing valuations in each world and then aggregating over the worlds:

Proposition 7. *Let $\mathfrak{I} = \langle \mathcal{I}, \models_{\mathcal{I}} \rangle$ be a possible world semantics, where $\mathcal{I} = \langle \langle W, R, \mathcal{M} \rangle, \nu_i \rangle$ for a finite W, and let $\mathsf{d}^{\mathcal{M}}_{\mathcal{G}}$ be a $\mathcal{G}$-dissimilarity generator for $\langle \Lambda_{\mathcal{M}}, \models_{\mathcal{M}} \rangle$ (e.g., of the form considered in Proposition 5). We define, for an aggregation function g, a function $\mathsf{d}^{g}_{\mathcal{G}}$, such that for any Γ and frame interpretations $I_1 = \langle \langle W, R, \mathcal{M} \rangle, \nu_1 \rangle$ and $I_2 = \langle \langle W, R, \mathcal{M} \rangle, \nu_2 \rangle \in \mathcal{I}$,*

$$\mathsf{d}^{g}_{\mathcal{G}}(\Gamma)(I_1, I_2) = g\big(\{\mathsf{d}^{\mathcal{M}}_{\mathcal{G}}(\Gamma)(\nu_1(w), \nu_2(w)) \mid w \in W\}\big)$$

where, for $i = 1, 2$, $\nu_i(w)$ is the 'restriction' of ν_i to the world w, that is: for every formula ψ, $\nu_i(w)(\psi) = \nu_i(w, \psi)$. Then:

a) $\mathsf{d}^{g}_{\mathcal{G}}$ is a $\mathcal{G}$-dissimilarity generator for $\mathfrak{I}$,
b) if $\mathcal{I}$ is $\mathcal{M}$-closed (that is, if for each $\langle \mathcal{F}, \nu \rangle \in \mathcal{I}$ and $\mu \in \Lambda_{\mathcal{M}}$ there is $\langle \mathcal{F}, \mu \rangle \in \mathcal{I}$), then $\mathsf{d}^{g}_{\mathcal{G}}$ is a normal $\mathcal{G}$-dissimilarity generator.

Example 10. A committee of three people, P_1, P_2 and P_3 should nominate two or less candidates among a and b for a governmental position. A committee member P_i may consult with any other member. Moreover, P_i will vote for $\mathsf{x} \in \{\mathsf{a}, \mathsf{b}\}$ only if P_i believes that x is qualified, and no member that P_i consults with believes otherwise. A candidate is recommended only upon a consensus.

A journalist j wants to predict the committee's recommendation, based on his partial knowledge about the committee and some leaking rumors. Suppose that he knows that P_1 believes that both candidates are qualified and that P_2 believes that b is qualified. Moreover, j knows that P_1 and P_2 consult with P_3, but P_3 never asks anyone else for advice.

This situation can be represented by the classical matrix and a modal language $\mathcal{L} = \{\Box, \wedge, \neg\}$. The atoms $\mathsf{q_a}$ and $\mathsf{q_b}$ respectively represent the belief that a and b are qualified, and the formula $\mathsf{Q_x} = \Box \mathsf{q_x}$ indicates that x is in the list of qualified candidates. Each world is associated with a committee member: $W = \{P_1, P_2, P_3\}$. Accessibility between worlds indicates a consulting relation between the members, thus: $R = \{\langle P_1, P_1 \rangle, \langle P_2, P_2 \rangle, \langle P_3, P_3 \rangle, \langle P_1, P_3 \rangle, \langle P_2, P_3 \rangle\}$. The corresponding frame is shown in Figure 2.

oarieli@mta.ac.il

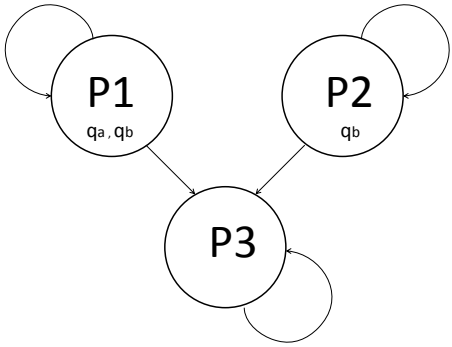


Fig. 2. The frame of Example 10

Next, suppose that two contradictory rumors are brought to j's attention: according to one, the list includes the names of both a and b. According to the other, at least one of the members disqualified a. This information may be represented by $\Gamma = \{\neg\Box\mathsf{q_a}, \Box\mathsf{q_a} \wedge \Box\mathsf{q_b}\}$. For maintaining this contradictory theory, j uses $\mid\!\sim_{\mathcal{S}}$, induced by the setting $\mathcal{S} = \langle \mathcal{G}^{\mathsf{At}}, \mathsf{d}^{\Sigma}, \Sigma\rangle$, where d^{Σ} is the dissimilarity-generator defined like in Proposition 7 for $g = \Sigma$ and $d^{\mathcal{M}}_{\mathcal{G}} = d^{\mathsf{h}}_{\mathcal{G}}$. Note that since $\mathfrak{I}$ is normal, by Theorem 1 and Proposition 7, $\mid\!\sim_{\mathcal{S}}$ is inconsistency-tolerant.

The dissimilarity calculations for the frame interpretations that correspond to the partial knowledge of j are given below (where $\psi^{\mathsf{i}}_{\mathsf{x}}$, for $\psi \in \{\mathsf{q}, \mathsf{Q}\}$, $\mathsf{x} \in \{\mathsf{a}, \mathsf{b}\}$ and $1 \leq i \leq 3$, denotes the value of the formula ψ_{x} in the world P_i).

	q^1_a	q^2_a	q^3_a	q^1_b	q^2_b	q^3_b	Q^1_a	Q^2_a	Q^3_a	Q^1_b	Q^2_b	Q^3_b	1	2	3
I_1	t	t	t	t	t	t	t	t	t	t	t	t	1	0	1
I_2	t	t	t	t	t	f	t	t	t	f	f	f	1	1	2
I_3	t	t	f	t	t	t	f	f	f	t	t	t	0	1	1
I_4	t	t	f	t	t	f	f	f	f	f	f	f	0	2	2
I_5	t	f	t	t	t	t	t	f	t	t	t	t	0	1	1
I_6	t	f	t	t	t	f	t	f	t	f	f	f	0	2	2
I_7	t	f	f	t	t	t	f	f	f	t	t	t	0	2	2
I_8	t	f	f	t	t	f	f	f	f	f	f	f	0	3	3

Legend. $1 = \mathsf{m}^{\Gamma}_{\mathcal{S}}(I_i, \neg\mathsf{Q}_a)$, $2 = \mathsf{m}^{\Gamma}_{\mathcal{S}}(I_i, \mathsf{Q}_a \wedge \mathsf{Q}_b))$, $3 = \mathsf{M}_{\mathcal{S}}(i, \Gamma)$.

Thus, $\Delta_{\mathcal{S}}(\Gamma) = \{I_1, I_3, I_5\}$, and so $\Gamma \mid\!\sim_{\mathcal{S}} \mathsf{Q_b}$ while $\Gamma \not\mid\!\sim_{\mathcal{S}} \mathsf{Q_a}$, $\Gamma \not\mid\!\sim_{\mathcal{S}} \neg\mathsf{Q_a}$. Based on his knowledge, then, j may assume that b will be nominated, while nothing can be predicted about a.

7 Conclusion

We have introduced a general method of supplementing different logics, based on denotational semantics, with extra apparatus assuring a proper tolerance of

oarieli@mta.ac.il

inconsistency, based on the notion of dissimilarity. As shown in [6], the properties of dissimilarities and of their generators guarantee that the logics that are obtained are inconsistency-tolerant. However, while the work in [6] investigates *specific* ways of defining dissimilarity-based logics for concrete forms of denotational semantics, here we provide a *generic and abstract* form of dissimilarity-based reasoning, suitable for *any type* of denotational semantics.

It should also be noted that, unlike many other frameworks of inconsistency handling, nothing is assumed here about the underlying semantics, apart of it being denotational. This, and the fact that dissimilarity is a more general notion than (pseudo) distances, imply that our approach may be used, e.g., for extending traditional distance-related methodologies in the context of revision and merging systems [17, 18, 21], cardinality-based methods for database repair [4, 9, 24], and forgetting-based approaches to reasoning with inconsistency [20, 22]. As a consequence, the standard two-valued propositional framework, which is the common platform for these methodologies, may be replaced by other non-classical, decidable platforms, involving many-valuedness, possible-worlds, and non-determinism.

An important subject for future research is a comparative study concerning the different entailment relations that are induced by different dissimilarity-based settings. This involves extensions to the non-monotonic case of works such as that in [3] (which introduces a list of desirable properties that paraconsistent consequence relations should have[5]).

References

[1] Arieli, O.: Distance-based paraconsistent logics. Journal of Approximate Reasoning 48(3), 766–783 (2008)

[2] Arieli, O., Avron, A.: General patterns for nonmonotonic reasoning: from basic entailments to plausible relations. Logic Journal of the IGPL 8(2), 119–148 (2000)

[3] Arieli, O., Avron, A., Zamansky, A.: Ideal paraconsistent logics. Studia Logica 99(1-3), 31–60 (2011)

[4] Arieli, O., Denecker, M., Bruynooghe, M.: Distance semantics for database repair. Annals of Mathematics and Artificial Intelligence 50(3-4), 389–415 (2007)

[5] Arieli, O., Zamansky, A.: Distance-based non-deterministic semantics for reasoning with uncertainty. Logic Journal of the IGPL 17(4), 325–350 (2009)

[6] Arieli, O., Zamansky, A.: Similarity-Based Inconsistency-Tolerant Logics. In: Janhunen, T., Niemelä, I. (eds.) JELIA 2010. LNCS (LNAI), vol. 6341, pp. 11–23. Springer, Heidelberg (2010)

[7] Avron, A., Lev, I.: Non-deterministic multi-valued structures. Journal of Logic and Computation 15, 241–261 (2005)

[8] Avron, A., Zamansky, A.: Non-deterministic multi-valued logics – A tutorial. In: Proc. ISMVL 2010, pp. 53–60. IEEE (2010)

[9] Bertossi, L.: Consistent query answering in databases. SIGMOD Records 35(2), 68–76 (2006)

[10] da Costa, N.C.A.: On the theory of inconsistent formal systems. Notre Dame Journal of Formal Logic 15, 497–510 (1974)

[5] Unlike our case, then, the work in [3] refers to *monotonic* logics (see Definition 1).

oarieli@mta.ac.il

[11] Dalal, M.: Investigations into a theory of knowledge base revision. In: Proc. AAAI 1988, pp. 475–479. AAAI Press (1988)
[12] de Amo, S., Carnielli, W.A., Marcos, J.: A Logical Framework for Integrating Inconsistent Information in Multiple Databases. In: Eiter, T., Schewe, K.-D. (eds.) FoIKS 2002. LNCS, vol. 2284, pp. 67–84. Springer, Heidelberg (2002)
[13] Fitting, M.: Many-valued modal logics. Fundamenta Informaticae 15(3-4), 235–254 (1991)
[14] Gabbay, D.: Theoretical foundation for non-monotonic reasoning. In: Proc. SCAI 1991, Part II. Structured Non-Monotonic Theories. IOS Press (1991)
[15] Gorogiannis, N., Hunter, A.: Implementing semantic merging operators using binary decision diagrams. Approximate Reasoning 49(1), 234–251 (2008)
[16] Kleene, S.C.: Introduction to Metamathematics. Van Nostrand (1950)
[17] Konieczny, S., Lang, J., Marquis, P.: Distance-based merging: a general framework and some complexity results. In: Proc. KR 2002, pp. 97–108. Morgan Kaufmann Publishers (2002)
[18] Konieczny, S., Pino Pérez, R.: Merging information under constraints: a logical framework. Journal of Logic and Computation 12(5), 773–808 (2002)
[19] Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
[20] Lang, J., Marquis, P.: Reasoning under inconsistency: A forgetting-based approach. Artificial Intelligence 174(12-13), 799–823 (2010)
[21] Lehmann, D., Magidor, M., Schlechta, K.: Distance semantics for belief revision. Journal of Symbolic Logic 66(1), 295–317 (2001)
[22] Lin, F., Reiter, R.: Forget it! In: Proc. AAAI Fall Symposium on Relevance, pp. 154–159 (1994)
[23] Lin, J., Mendelzon, A.O.: Knowledge base merging by majority. In: Dynamic Worlds: From the Frame Problem to Knowledge Management. Kluwer (1999)
[24] Lopatenko, A., Bertossi, L.: Complexity of Consistent Query Answering in Databases Under Cardinality-Based and Incremental Repair Semantics. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 179–193. Springer, Heidelberg (2006)
[25] Urquhart, A.: Many-valued logic. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. II, pp. 249–295. Kluwer (2001)

oarieli@mta.ac.il

Revising Belief without Revealing Secrets*

Joachim Biskup and Cornelia Tadros

Technische Universität Dortmund, Germany
{joachim.biskup,cornelia.tadros}@cs.tu-dortmund.de

Abstract. In multiagent systems, agents interact and in particular exchange information to achieve a joint goal, e.g., arrange a meeting, negotiate a sales contract etc. An agent, as a rational reasoner, is able to incorporate new information into her belief about her environment (belief revision) or to share her belief with other agents (query answering). Yet, such an agent might be interested to hide confidential parts of her belief from other negotiating agents while these agents are supposed to reason about her reactions to revisions and queries. We study how an agent can control her reactions to revisions and queries requested by another agent who may attempt to skeptically entail confidential beliefs. As our main contribution, we present procedures that provably enforce confidentiality, to be employed by the reacting agent.

Keywords: Confidential Belief, Belief Revision, Multiagent System, Inference Control, Skeptical Entailment.

1 Introduction

1.1 Problem Statement

As commonly known by the database security community, preventing only direct access to sensitive data items might not suffice to ensure confidentiality of the information represented by these data. Rather, e.g., after accessing non-protected data items by query requests, a database user might be able to deduce sensitive information from the accessed data. Moreover, such a deduction might exploit the user's knowledge of database constraints and further a priori knowledge. The challenge to prevent such deductions is known as the *inference problem*. *Controlled Query Evaluation* (CQE) presents solutions to the inference problem for query-answering in several types of database systems, including relational databases and incomplete propositional databases, and is extended to database and view updates in [5], cf. [3] for a survey.

Beyond classical database systems, in this article we deal with the following version of the inference problem: how can an agent *revise* and *release* parts of her *belief* during the interaction with another agent without compromising confidentiality of sensitive parts of her belief. In this context, an agent is seen as

* This work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project A5.

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 51–70, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

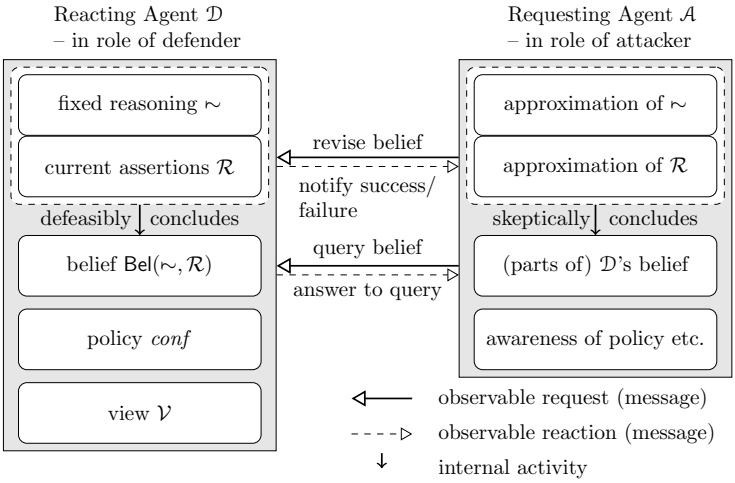


Fig. 1. Focussed scenario of two agents

an autonomous computing system that interacts with her environment directed by internal goals [18]. The environment often includes other agents (multiagent system). An agent's decisions are guided by her belief about her environment while her belief is usually incomplete and based on defeasible conclusions and, thus, maybe revised during an interaction due to new evidence, cf. [1]. Moreover, in order to adapt her belief to changes in her environment, the agent updates her belief, cf. [15,1], but for the time being, we do not study confidentiality preserving updates. In line with CQE, Biskup et al [6] list requirements for confidentiality preserving negotiations in multiagent systems and, based on these requirements, suggest a BDI architecture but exhibiting neither a concrete reasoning component to derive belief nor procedures for controlling the related belief operations. In this article, we refine their work by adapting the model of [1,15] as the reasoning component and then showing how queries and revisions can be effectively controlled to preserve confidentiality based on this model.

1.2 Scenario of Two Interacting Agents

We consider a basic scenario where an agent $\mathcal{A}$ requests queries about another agent $\mathcal{D}$'s belief or requests revisions of $\mathcal{D}$'s belief while agent $\mathcal{D}$ reacts to these requests in return. This scenario is outlined by Fig. 1 and explained in the following. Basically, the agents are supposed to interact to achieve a joint goal, e.g., treatment of a patient, arrangement of a meeting [6] etc. Here, agent $\mathcal{A}$ may be a nurse giving reports about a patient to agent $\mathcal{D}$, a physician, (revision request) or taking instructions for the patient's treatment (query request).

As a means of communication, the agents share propositional logic over the same countably infinite alphabet $\mathcal{A}$ as a common basic language. Here, $\mathcal{L}_{pl}$ denotes the set of all propositional formulas over the alphabet $\mathcal{A}$ of atomic

oarieli@mta.ac.il

propositions. Within the basic scenario we focus on the following fixed roles of the agents. On the one hand, $\mathcal{D}$ is a reasoner about her environment in general with defeasible reasoning $\mathrel{|\!\sim}$ and as such forms belief about this environment. $\mathcal{D}$'s reasoning supports the usual and desired functionality of an interacting agent. On the other hand, $\mathcal{A}$ is a reasoner about $\mathcal{D}$'s belief with skeptical reasoning $\mathsf{skeptical}$. $\mathcal{A}$'s reasoning results from curiosity about $\mathcal{D}$'s belief. Certainly, implementing a multiagent system, each implemented agent should be equipped at least with $\mathcal{D}$'s capabilities but such an implementation would additionally require confidentiality preserving initiation of requests which we do not cover in this article (cf. Sect. 3).

1.3 Outline of Agent $\mathcal{D}$'s Belief

The internal components of $\mathcal{D}$ in Fig. 1 are designed as follows, cf. [1,15,10]. $\mathcal{D}$ faces generally incomplete information (*current assertions* $\mathcal{R} \subseteq \mathcal{L}_{pl}$) about her environment, e.g., a patient's present condition. Moreover, to guide her decisions, $\mathcal{D}$ must be capable to draw reasonable conclusions from the available information $\mathcal{R}$. This capability is modeled by a *consequence relation* (*fixed reasoning* $\mathrel{|\!\sim} \subseteq \mathcal{L}_{pl} \times \mathcal{L}_{pl}$) that maps assertions to conclusions where we read $A \mathrel{|\!\sim} B$ as "agent $\mathcal{D}$ *defeasibly concludes* B from A". Gathering all conclusions from her current assertions, $\mathcal{D}$ forms her *belief* $\mathsf{Bel}(\mathrel{|\!\sim}, \mathcal{R})$ about the environment:

$$\mathsf{Bel}(\mathrel{|\!\sim}, \mathcal{R}) := \{B \in \mathcal{L}_{pl} \mid \mathsf{con}(\mathcal{R}) \mathrel{|\!\sim} B\} \text{ where } \mathsf{con}(\mathcal{R}) := \begin{cases} \bigwedge_{F \in \mathcal{R}} F & \text{if } \mathcal{R} \neq \emptyset, \\ \mathit{true} & \text{otherwise.} \end{cases}$$

During an interaction, $\mathcal{A}$ may add some information $A \in \mathcal{L}_{pl}$ to $\mathcal{D}$'s current assertions $\mathcal{R}$ in requesting a belief revision by A from $\mathcal{D}$. Agent $\mathcal{D}$ might reject to consider this additional information due to a conflict with some *unquestionable belief* so that the revision will fail in this case. Otherwise, in the light of this additional information, $\mathcal{D}$ might reconsider and refute some of her conclusions while reasoning with $\mathrel{|\!\sim}$. For example, as a physician, agent $\mathcal{D}$ might diagnose cancer if she finds out in an examination that the patient shows symptom x (represented by $x \mathrel{|\!\sim} c$ where x denotes "the patient shows symptom x" and c denotes "the patient has cancer"). However, later she might withdraw her diagnosis if agent $\mathcal{A}$ reports to her another examination that indicates symptom y (represented by $x \wedge y \not\mathrel{|\!\sim} c$). Here, we denote the complement $(\mathcal{L}_{pl} \times \mathcal{L}_{pl}) \setminus \mathrel{|\!\sim}$ of the consequence relation $\mathrel{|\!\sim}$ by $\not\mathrel{|\!\sim}$. The reconsideration of previous conclusions (belief) at the arrival of new information is known as *belief revision* (see, e.g., [10]).

As a special case, agent $\mathcal{D}$ may employ a classical database, e.g., $\mathcal{R}$ may be a complete [4] or incomplete propositional database [8]. Then, $\mathrel{|\!\sim}$ corresponds either to the propositional model-of operator, which evaluates (propositional) formulas by a truth-value assignment – the complete database, or the propositional entailment operator by which $\mathcal{D}$ draws conclusions from a set of (propositional) formulas – the incomplete database.

oarieli@mta.ac.il

1.4 Preserving Confidentiality of Belief towards Agent $\mathcal{A}$

Under the confidentiality aspect, however, agent $\mathcal{D}$ usually is obliged to keep sensitive personal information, e.g., regarding the patient, confidential or has own confidentiality interests. But agent $\mathcal{D}$ can only hide her own belief and has no impact on how possibly sensitive information is disseminated by other sources. For example, although as a physician agent $\mathcal{D}$ might desire to hide the *fact* that the patient has cancer, $\mathcal{D}$ can only aim at hiding her own belief in the diagnosis cancer. For this purpose, agent $\mathcal{D}$ declares a fixed *confidentiality policy* $conf \subset_{fin} \mathcal{L}_{pl}$. If $\mathcal{D}$ believes B with $B \in conf$ then agent $\mathcal{D}$ aims to hide this fact from agent $\mathcal{A}$. Continuing our previous example, with $conf = \{c\}$ agent $\mathcal{D}$ aims to conceal the fact $c \in \mathsf{Bel}(\sim, \{x\})$ from agent $\mathcal{A}$ whereas $c \notin \mathsf{Bel}(\sim, \{x, y\})$ may be revealed. We call each assertion $B \in conf$ a *confidential belief* regardless of whether or not agent $\mathcal{D}$ actually believes B. Note that the propositional language for policy declaration cannot express a purpose of hiding a non-belief, e.g., hiding that agent $\mathcal{D}$ cannot reach a diagnosis of cancer.

From the confidentiality perspective, agent $\mathcal{D}$ holds the *role of a defender* against agent $\mathcal{A}$ in the *role of a (potential) attacker* (the latter in attempting to obtain confidential information against $\mathcal{D}$'s obligations or interests). In this article, agent $\mathcal{D}$ should guarantee confidentiality as the following property, which formalizes the semantics of the policy $conf$, cf. [6]:

> Starting with any initial situation of $\mathcal{D}$ (including policy $conf$),
> after any finite interaction between $\mathcal{A}$ and $\mathcal{D}$,
> for each confidential belief $B \in conf$,
> by her observations of $\mathcal{D}$'s reactions $\mathcal{A}$ cannot distinguish the actual situation from an alternative situation in which $\mathcal{D}$ does not believe B.

$\mathcal{D}$'s opponent, agent $\mathcal{A}$, aims to actually determine what $\mathcal{D}$ believes in each of these indistinguishable situations or, in other words, what $\mathcal{D}$ certainly believes. Because $\mathcal{D}$ derives her belief from her current assertions $\mathcal{R}$ by her fixed reasoning $\sim$, $\mathcal{A}$ consequently tries to approximate these two components in accordance with the observed reactions from $\mathcal{D}$. From these approximations $\mathcal{A}$ may conclude parts of $\mathcal{D}$'s belief with certainty (i.e., skeptically conclude these parts):

- $\mathcal{A}$ approximates $\mathcal{D}$'s fixed reasoning $\sim$
  (1) by a set $\mathcal{B}^+ \subset_{fin} \mathcal{L}_{pl} \times \mathcal{L}_{pl}$ that describes $\mathcal{D}$'s observed behavior to draw conclusions. In a special case, $\mathcal{B}^+$ approximates $\sim$ from below: $\mathcal{B}^+ \subseteq \sim$.
  (2) by a set $\mathcal{B}^- \subset_{fin} \mathcal{L}_{pl} \times \mathcal{L}_{pl}$ that describes $\mathcal{D}$'s observed behavior not to draw conclusions. In a special case, $\mathcal{B}^-$ approximates $\sim$ from above: $\mathcal{B}^- \subseteq \not\sim$.

- $\mathcal{A}$ approximates $\mathcal{D}$'s current assertions $\mathcal{R}$ by a set $\mathcal{C} \subseteq \mathcal{L}_{pl}$ that describes which of the assertions propositionally entailed by $\mathcal{R}$ are visible to $\mathcal{A}$.
- $\mathcal{A}$ draws conclusions by a *skeptical entailment operator*:

$$\mathsf{skeptical}(\mathcal{B}^+, \mathcal{B}^-, \mathcal{C}) = \{B \in \mathcal{L}_{pl} \mid \text{for each consequence relation } \sim' \text{ possible under } \mathcal{B}^+, \mathcal{B}^- \text{ and } \mathcal{C}: \mathsf{con}(\mathcal{C}) \sim' B\}.$$

oarieli@mta.ac.il

Informally, $\mathcal{A}$ considers a consequence relation possible iff this relation agrees with her approximations. To effectively control her reactions, agent $\mathcal{D}$ needs $\mathcal{A}$'s approximations at the beginning and during an interaction so that $\mathcal{D}$ can simulate $\mathcal{A}$'s skeptical reasoning and detect whether a planned reaction would allow $\mathcal{A}$ to skeptically conclude a confidential belief. For this purpose, $\mathcal{D}$ maintains a *view* $\mathcal{V} = \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ that contains $\mathcal{A}$'s presumable approximations.

Assumption 1. *$\mathcal{D}$'s view component $\mathcal{V}$ and $\mathcal{A}$'s approximation coincide.*

1.5 Organization of Article

In the remainder, we will elaborate on the scenario of the previous sections but we decide to represent $\mathcal{D}$'s fixed reasoning by ordinal conditional functions (OCFs) [16]. In future work, we may consider different assumptions about $\mathcal{D}$'s fixed reasoning. However, the representation by OCFs is very general. In particular, propositional conditional logic cannot distinguish between OCFs and possibility measures [12] so that we can argue that our results still hold for a representation by possibility measures. Sect. 2 introduces ordinal conditional functions. In Sect. 3 we describe the agents' interaction.

Our original main achievements are presented in the following sections:
In Sect. 4, we describe the capabilities of the potential attacker agent $\mathcal{A}$ in detail, in particular her skeptical entailment operator. Within this article, we focus on defining and investigating this entailment operator in terms of the model of agents and observations previously outlined. However, we postpone to choose an established logic and, then, to characterize $\mathcal{A}$'s skeptical entailment in this logic to further research. Our way of describing the potential attacker $\mathcal{A}$ is motivated by two reasons. First, we argue that an appropriate choice of an established logic should be based on an already well-understood semantic model of $\mathcal{A}$'s capabilities and potential behavior. Second, such a choice depends on the representation of $\mathcal{D}$'s fixed reasoning so that a choice at this early stage of our work would make it more difficult to consider different representations in future work.

In Sect. 5, we outline the defender $\mathcal{D}$'s design under confidentiality requirements based on [6]. In particular, we equip agent $\mathcal{D}$ with procedures to control her reactions to queries and revisions and we formally prove that by means of these procedures $\mathcal{D}$ preserves confidentiality throughout the interaction with agent $\mathcal{A}$.

We conclude with Sect. 6 in which we discuss our results (also in comparison to classical databases) and compare them to related work.

2 Agent $\mathcal{D}$'s Belief

To operate, agent $\mathcal{D}$ relies on her belief $\mathsf{Bel}(\mathsf{r}\!\sim, \mathcal{R})$ about her environment. The environment itself may be ideally mapped to a truth-value assignment of all atomic propositions in $\mathcal{A}$. Each truth-value assignment thus describes some state of the environment and is viewed as a *world*. Although agent $\mathcal{A}$ is part of the environment, for simplicity, agent $\mathcal{D}$ does not consider $\mathcal{A}$'s reasoning in her belief $\mathsf{Bel}(\mathsf{r}\!\sim, \mathcal{R})$ but simulates $\mathcal{A}$'s reasoning separately (Sect. 5.2).

oarieli@mta.ac.il

Example 1. In our running example, we consider the countably infinite alphabet $\mathcal{A} = \{s_1, s_2, s_3, s_4, c, \ldots\}$ where s_i stands for "the patient shows symptom s_i" and c for "the patient has cancer". We write down a truth-value assignment over $\mathcal{A}$ only for the relevant atoms where, e.g. , $s_1 \neg s_2\ s_3\ s_4 \neg c$ denotes that s_1, s_3 and s_4 are assigned to *true* whereas s_2 and c are assigned to *false*.

We define $\mathcal{D}$'s fixed reasoning $\mathrel{|\!\sim}$ by an ordinal conditional function (OCF). This representation gives the following intuition behind $\mathcal{D}$'s process of reasoning: $\mathcal{D}$ orders worlds by a "degree of surprise" (rank) when finding out that a world describes the state of her environment, ranging from 0 "no surprise" to ∞ "totally surprising". Worlds with a lower degree of surprise are more plausible to $\mathcal{D}$.

Definition 1 (OCF, Rank of Formulas [1,16]). *Let Ω be the set of all truth-value assignments (worlds) over the alphabet $\mathcal{A}$. Then, an OCF is a function $\kappa : \Omega \to \mathbb{N}_0 \cup \{\infty\}$ with $\kappa^{-1}(0) \neq \emptyset$.*[1] *It defines the rank of a formula $A \in \mathcal{L}_{pl}$ by*

$$\kappa(A) := \begin{cases} \min\limits_{\omega \models_{pl} A} \kappa(\omega) & \text{if } A \text{ is satisfiable,} \\ \infty & \text{otherwise,} \end{cases}$$

where $\models_{pl}$ denotes the model-of operator in propositional logic.

By assigning a world ω the rank ∞, agent $\mathcal{D}$ states that this world impossibly describes the state of her environment so that κ partitions Ω into possible worlds ($\kappa(\omega) < \infty$) and impossible worlds ($\kappa(\omega) = \infty$). Moreover, we can say that by $\kappa(\neg B) = \infty$ agent $\mathcal{D}$ *unquestionably believes* (i.e., knows) a formula B. We will further clarify the meaning of "$\mathcal{D}$ unquestionably believes" throughout this section. With the ordering κ of worlds in mind, $\mathcal{D}$ defeasibly concludes a formula B from a formula A ($A \mathrel{|\!\sim}_\kappa B$) if all models ω of A that are most plausible under κ justify a conclusion of B, i.e., for all such ω we have $\omega \models_{pl} A \wedge B$ (or equivalently: there is no most plausible model ω' of A that objects to concluding B, i.e., $\omega' \models_{pl} A \wedge \neg B$):

Definition 2 (Consequence Relation $\mathrel{|\!\sim}_\kappa$). *Let κ be an OCF over Ω and $A, B \in \mathcal{L}_{pl}$. Then, $A \mathrel{|\!\sim}_\kappa B$, B is a defeasible conclusion from A w.r.t. κ iff*

- *there is no most plausible model of A under κ:*
  $\kappa(A) = \infty$, or
- *all most plausible models ω of A under κ satisfy B:*
  $\kappa(A) \neq \infty$ and for all $\omega \in \Omega$ s.t. $\omega \models_{pl} A$ and $\kappa(\omega) = \kappa(A)$ holds $\omega \models_{pl} B$.

In the following proposition, we see that "$\mathcal{D}$ unquestionably believes B" means that $\mathcal{D}$ will conclude B from any assertion A, and that unquestionable belief can be characterized by $\mathrel{|\!\sim}_\kappa$:

Proposition 1 (Properties of $\mathrel{|\!\sim}_\kappa$).[2] *Let κ be an OCF and $B \in \mathcal{L}_{pl}$. Then, the consequence relation $\mathrel{|\!\sim}_\kappa$ has the following properties:*

[1] Like in [1], we simplified the original definition of [16].

oarieli@mta.ac.il

1. *If* $\kappa(\neg B) = \infty$*, then for all* $A \in \mathcal{L}_{pl}$: $A \mathrel{|\!\sim}_\kappa B$.
2. $\kappa(\neg B) = \infty$ *iff* $\neg B \mathrel{|\!\sim}_\kappa$ *false*.

$\mathcal{D}$'s fixed reasoning and her current assertions are jointly represented by a belief base and define her current belief:

Definition 3 (Belief Base, Belief, cf. [15]). *Let* κ *be an OCF and* $\mathcal{R} \subset_{fin} \mathcal{L}_{pl}$ *a finite set of propositions such that* $\kappa(\mathsf{con}(\mathcal{R})) \neq \infty$*. Then,*

- *the pair* $\langle \kappa, \mathcal{R} \rangle$ *is called a belief base;*
- *the set* $\mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle) = \{A \in \mathcal{L}_{pl} \mid \mathsf{con}(\mathcal{R}) \mathrel{|\!\sim}_\kappa A\}$ *is called belief.*

Proposition 2 (Properties of $\mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle)$). [2] *Let* $\langle \kappa, \mathcal{R} \rangle$ *be a belief base. Then,* $\mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle)$ *has the following properties:*

- *The set* $\mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle)$ *is closed under propositional entailment* $\vdash_{pl}$.
- $\mathcal{R} \subseteq \mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle)$.
- *If* $\kappa(\neg B) = \infty$*, then* $B \in \mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle)$.

Example 2. We continue our running example. Agent $\mathcal{D}$'s fixed reasoning is represented by κ_1 as follows:

∞	$*$	s_2	s_3	$*$	$*$	l_∞												
2	s_1	$\neg s_2$	s_3	s_4	c	l_{21}	s_1	$\neg s_2$	$\neg s_3$	s_4	c	l_{22}	s_1	s_2	$\neg s_3$	s_4	c	l_{23}
1	s_1	$\neg s_2$	s_3	$*$	$\neg c$	l_{11}	s_1	$\neg s_2$	$\neg s_3$	$*$	$\neg c$	l_{12}	s_1	s_2	$\neg s_3$	$*$	$\neg c$	l_{13}
0	s_1	$\neg s_2$	s_3	$\neg s_4$	c	l_{01}	s_1	$\neg s_2$	$\neg s_3$	$\neg s_4$	c	l_{02}	s_1	s_2	$\neg s_3$	$\neg s_4$	c	l_{03}
	$\neg s_1$	$\neg s_2$	s_3	$*$	$*$	l_{04}	$\neg s_1$	$\neg s_2$	$\neg s_3$	$*$	$*$	l_{05}	$\neg s_1$	s_2	$\neg s_3$	$*$	$*$	l_{06}

Here $*$ stands for an arbitrary assignment of the corresponding atom and each assignment has a label l_{ij} on its right for later reference. $\mathcal{D}$ unquestionably believes symptoms s_2 and s_3 not to occur at the same time because $\kappa_1(s_2 \wedge s_3) = \infty$ (see l_∞). Thus by Prop. 1, $\mathcal{D}$ reasons for any assertion $A \in \mathcal{L}_{pl}$ that $A \mathrel{|\!\sim}_{\kappa_1} \neg s_2 \vee \neg s_3$. By Def. 2, to draw conclusions from, say, s_3, $\mathcal{D}$ only inspects the most plausible models of s_1 (which are given by l_{01} and l_{04}). Following Def. 2, agent $\mathcal{D}$ neither diagnoses cancer $s_3 \not\mathrel{|\!\sim}_{\kappa_1} c$ (see l_{04}) nor excludes such a diagnosis $s_3 \not\mathrel{|\!\sim}_{\kappa_1} \neg c$ (see l_{04}). With the additional information $s_1 \vee s_2$, $\mathcal{D}$ diagnoses cancer: $s_3 \wedge (s_1 \vee s_2) \mathrel{|\!\sim}_{\kappa_1} c$ (see l_{01}).

3 Uncontrolled Agent Interaction

In this article, we study interactions in which only two agents participate, $\mathcal{D}$ (in the role of a defender) and $\mathcal{A}$ (in the role of a potential attacker). Such interactions can be viewed as a private conversation between $\mathcal{A}$ and $\mathcal{D}$ amongst a group of more than two agents.

[2] Proposition 1 and 2 follow from or correspond to well-known results from the literature on the properties of $\mathrel{|\!\sim}_\kappa$, cf. [12] for selected results and cf. [14] for a discussion of unquestionable belief and a converse approach.

oarieli@mta.ac.il

Assumption 2. *Communication acts are visible to all agents. Communicated data is only readable by the intended recipients of the data.*

In our example, agent $\mathcal{A}$, a nurse, can find out whether agent $\mathcal{D}$ received reports about the patient from other staff but cannot read these reports – including the type of examination, laboratory tests involved etc. As a consequence of Assumption 2, agent $\mathcal{A}$ is aware that agent $\mathcal{D}$ does not communicate with other agents in between. In our scenario, agent $\mathcal{D}$ only reacts to requests from agent $\mathcal{A}$ (reactive part), leaving further interaction among more than two agents and requests initiated by $\mathcal{D}$ (active part) for future work, cf. [6]. Moreover we focus on the case that agent $\mathcal{A}$ can only request the following from $\mathcal{D}$ where $A \in \mathcal{L}_{pl}$:

– $\texttt{que}(A)$ (query about current belief of A): reply to "$A \in \mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle)$?".
– $\texttt{rev}(A)$ (belief revision by A): add A to the current assertions $\mathcal{R}$.

On a query request $\texttt{que}(A)$, without controlling her reactions for the sake of confidentiality, agent $\mathcal{D}$ returns the result of the ordinary evaluation

$$\texttt{eval}(A)(\langle \kappa, \mathcal{R} \rangle) := \begin{cases} \textit{true} & \text{if } A \in \mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle), \\ \textit{false} & \text{if } \neg A \in \mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle), \\ \textit{undef} & \text{otherwise.} \end{cases}$$

By a revision request $\texttt{rev}(A)$, agent $\mathcal{A}$ provides $\mathcal{D}$ with information about an environment that does not evolve during the interaction. Without controlling her reaction for the sake of confidentiality, on the request $\texttt{rev}(A)$ agent $\mathcal{D}$ first checks whether she unquestionably believes that the assertions A and $\mathcal{R}$ cannot hold at the same time, i.e., she considers the conjunction of these assertions impossible in κ. Only if the conjunction of the assertions is possible, $\mathcal{D}$ sets her belief base to the result of the operation $\langle \kappa, \mathcal{R} \rangle \circ \{A\}$:

Definition 4 (Revision $\circ$, cf. [1]). *Let $\langle \kappa, \mathcal{R} \rangle$ be a belief base and $A \in \mathcal{L}_{pl}$, Then, revision of $\langle \kappa, \mathcal{R} \rangle$ by A is defined as*

$$\langle \kappa, \mathcal{R} \rangle \circ \{A\} = \begin{cases} \langle \kappa, \mathcal{R} \cup \{A\} \rangle & \textit{if } \kappa(\mathsf{con}(\mathcal{R}) \wedge A) \neq \infty, \\ \langle \kappa, \mathcal{R} \rangle & \textit{otherwise.} \end{cases}$$

Finally, $\mathcal{D}$ reacts by notifying $\mathcal{A}$ whether $\mathcal{D}$'s belief base has been revised as follows:

$$\texttt{notify}(A)(\langle \kappa, \mathcal{R} \rangle) := \begin{cases} \textit{true} \text{ (Success)} & \text{if } \kappa(\mathsf{con}(\mathcal{R}) \wedge A) \neq \infty, \\ \textit{false} \text{ (Failure)} & \text{otherwise.} \end{cases}$$

Here, we chose to let $\mathcal{D}$ throw away the new information A in case of a failure to simplify confidentiality preservation. With this approach, $\mathcal{D}$'s notification to $\mathcal{A}$ after a revision can only reveal whether $\mathcal{D}$ considers $\mathsf{con}(\mathcal{R}) \wedge A$ possible or not. Generally, $\mathcal{D}$ should merge the new information with the previously gathered information to obtain a set of possible assertions (similar to repairing inconsistency), e.g., as suggested by [10]. With merging, more complicated notifications about results must be controlled resulting in more involved computations.

oarieli@mta.ac.il

By an update request, not further considered in this article, agent $\mathcal{A}$ can report a change of the environment to agent $\mathcal{D}$, e.g., changes in the patient's condition due to the patient's treatment. An update may also affect $\mathcal{D}$'s reasoning $\sim$ so that $\mathcal{D}$'s reasoning would no longer be fixed, cf. [15,1]. Confidentiality preserving updates have been proposed in the context of propositional complete databases (cf. [5] for a lying and [7] for a refusal approach). We leave the extension of these work to the more general context of belief bases for future investigation.

In Sect. 4, the requests sent and the reactions observed are agent $\mathcal{A}$'s starting point for approximating agent $\mathcal{D}$'s reasoning:

Definition 5 (History $\mathcal{H}$ of an Interaction). *A history $\mathcal{H}$ of an interaction is a finite sequence of pairs of request and respective reaction. These pairs have the form* $\begin{cases} (\mathtt{que}(A), react_{\mathtt{que}}) & \text{with } react_{\mathtt{que}} \in \{true, false, undef\}, A \in \mathcal{L}_{pl}, \\ (\mathtt{rev}(A), react_{\mathtt{rev}}) & \text{with } react_{\mathtt{rev}} \in \{true, false\}. \end{cases}$

Example 3. $\mathcal{D}$'s belief about a particular patient is represented by the belief base $\langle \kappa_1, \{s_3\} \rangle$, cf. Ex. 2. Now, $\mathcal{A}$ asks about a diagnosis of cancer by the query $\mathtt{que}(c)$ to which $\mathcal{D}$ replies with *undef*. Receiving an additional report by $\mathcal{A}$ indicating symptom s_1 or s_2 via the request $\mathtt{rev}(s_1 \vee s_2)$, $\mathcal{D}$ now diagnoses cancer. Now, $\mathcal{D}$ answers the query $\mathtt{que}(c)$ with *true*. The history of the whole interaction is $\langle (\mathtt{que}(c), undef), (\mathtt{rev}(s_1 \vee s_2), true), (\mathtt{que}(c), true) \rangle$.

If $\mathcal{D}$ was thereafter informed that the patient shows symptom s_4, the diagnosis would be $\neg c \in \mathsf{Bel}(\langle \kappa_1, \{s_3, s_1 \vee s_2, s_4\} \rangle)$ (see l_{11}).

4 $\mathcal{A}$'s View on the Uncontrolled Belief Revision Process

Agent $\mathcal{D}$ can only enforce and guarantee confidentiality (Sect. 5) under assumptions about $\mathcal{A}$'s – the potential attacker's – capabilities and behavior.

Assumption 3. *Agent $\mathcal{A}$ is aware of agent $\mathcal{D}$'s design, especially the design of her reasoning component and the code of procedures (Sect. 5.2) as well as the definitions of functions and operators (Sect. 3) applied by $\mathcal{D}$. Moreover, $\mathcal{A}$ may read the components conf and $\mathcal{V}$ any time during the interaction.*

This assumption prevents to achieve "security by obscurity" where obscurity means relying on $\mathcal{A}$'s ignorance of $\mathcal{D}$'s design and the policy *conf*. As for the latter, $\mathcal{A}$ could guess or even know (parts of) the confidentiality policy, e.g., c (cancer) is certainly confidential (cf. [3,4] for approaches that hide the policy).

Assumption 4. *Agent $\mathcal{A}$ is a skeptical reasoner (Def. 8) who has unlimited computing power for the computation of skeptical entailment.*

In the following subsections, we characterize $\mathcal{A}$'s skeptical reasoning in the two converse situations when either $\mathcal{A}$ sees all of $\mathcal{D}$'s current assertions or $\mathcal{D}$ has hidden parts thereof from $\mathcal{A}$.

oarieli@mta.ac.il

4.1 Skeptical Reasoning with Visible Current Assertions

Essentially, due to the open design (Assumption 3), during an uncontrolled interaction $\mathcal{A}$ is aware that $\mathcal{D}$ replies to a query $\mathtt{que}(A)$ with *true* iff $\mathcal{D}$ defeasibly concludes A from her current assertions $\mathcal{R}$ ($\mathsf{con}(\mathcal{R}) \mathrel{|\!\sim}_\kappa A$, Def. 3). This interrelation helps $\mathcal{A}$ to approximate $\mathcal{D}$'s fixed reasoning $\mathrel{|\!\sim}_\kappa$. The same argument applies when $\mathcal{A}$ is notified by $\mathcal{D}$ that a revision $\mathtt{rev}(A)$ has failed. In this situation, $\mathcal{A}$ is aware that $\mathcal{D}$ unquestionably believes $\neg A$ in the context of her current assertions $\mathcal{R}$ ($\mathsf{con}(\mathcal{R}) \wedge A \mathrel{|\!\sim}_\kappa \mathit{false}$, Prop. 1).

In the same manner, each of $\mathcal{D}$'s reactions helps $\mathcal{A}$ to approximate $\mathcal{D}$'s reasoning as detailed in the following. Yet, in order to definitely follow $\mathcal{D}$'s reasoning from her current assertions $\mathcal{R}$, agent $\mathcal{A}$ needs to be certain about $\mathcal{R}$. To start with, we assume that the assertions $\mathcal{R}$ are visible to $\mathcal{A}$. Under this assumption, $\mathcal{A}$ can easily approximate $\mathcal{D}$'s fixed reasoning $\mathrel{|\!\sim}_\kappa$ as illustrated by the following example and implemented by Procedures 1 and 2 (where $\mathcal{C} = \mathcal{R}$ by assumption).

Example 4. Reviewing the interaction between $\mathcal{A}$ and $\mathcal{D}$ from Ex. 3, we see that $\mathcal{A}$ finds out the following from $\mathcal{D}$'s reactions to her requests:

Answer *undef* to $\mathtt{que}(c)$:	$s_3 \not\mathrel{|\!\sim}_{\kappa_1} c$ and $s_3 \not\mathrel{|\!\sim}_{\kappa_1} \neg c$
Notification of success after $\mathtt{rev}(s_1 \vee s_2)$:	$s_3 \wedge (s_1 \vee s_2) \not\mathrel{|\!\sim}_{\kappa_1} \mathit{false}$
Answer *true* to $\mathtt{que}(c)$:	$s_3 \wedge (s_1 \vee s_2) \mathrel{|\!\sim}_{\kappa_1} c$

$\mathcal{A}$'s approximations are represented and summarized by the following definition.

Definition 6 (View $\mathcal{V}$). *Let $\mathcal{B}^+, \mathcal{B}^- \subset_{fin} \mathcal{L}_{pl} \times \mathcal{L}_{pl}$ be finite relations and $\mathcal{C} \subset_{fin} \mathcal{L}_{pl}$ be a finite set of formulas. Then, a view $\mathcal{V}$ is the triple $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$. We abbreviate $\mathcal{B} = \mathcal{B}^+ \cup \mathcal{B}^-$.*

Surely, $\mathcal{A}$ expects that there is at least one consequence relation that, together with the visible assertions $\mathcal{C}$, explains $\mathcal{D}$'s reactions described by $\mathcal{B}$ (i.e., there exists a possible consequence relation under $\mathcal{V}$ (Def. 7, below)). The final view after the interaction in Ex. 4 is

$$\langle \{(s_3 \wedge (s_1 \vee s_2), c)\},\ \{(s_3, c),\ (s_3, \neg c),\ (s_3 \wedge (s_1 \vee s_2), \mathit{false})\},\ \{s_3, s_1 \vee s_2\} \rangle.$$

Before the interaction, agent $\mathcal{A}$ can have an a priori view $\mathcal{V}_0$, e.g., as a nurse, $\mathcal{A}$ may be aware of some of $\mathcal{D}$'s medical expertise ($\mathcal{B}$)[3] or possible lack of expertise ($\mathcal{B}^-$), and may see some reports regarded by $\mathcal{D}$ about the particular patient ($\mathcal{C}$). Primarily, by her approximations an attacking agent $\mathcal{A}$ aims to conclude with certainty that $\mathcal{D}$ currently believes a confidential belief $B \in \mathit{conf}$ (Sect. 1). Before we move to defining $\mathcal{A}$'s reasoning by a skeptical entailment operator we give the intuition behind it: Due to the open design, $\mathcal{A}$ is aware that $\mathcal{D}$'s fixed reasoning is a consequence relation defined by some OCF. Moreover, $\mathcal{A}$

[3] In the case of $\mathcal{B}^-$, $\mathcal{D}$ might show medical expertise by rejecting inappropriate conclusions.

oarieli@mta.ac.il

Procedure 1. app_by_que

Input: query A, query answer *ans*, view $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$
Output: refined view
1: **if** $ans = true$ **then return** $\langle \mathcal{B}^+ \cup \{(\mathsf{con}(\mathcal{C}), \ A)\}, \ \mathcal{B}^-, \mathcal{C} \rangle$
2: **else if** $ans = false$ **then return** $\langle \mathcal{B}^+ \cup \{(\mathsf{con}(\mathcal{C}), \neg A)\}, \ \mathcal{B}^-, \mathcal{C} \rangle$
3: **else return** $\langle \mathcal{B}^+, \ \mathcal{B}^- \cup \{(\mathsf{con}(\mathcal{C}), A), (\mathsf{con}(\mathcal{C}), \neg A)\}, \ \mathcal{C} \rangle$

Procedure 2. app_by_rev

Input: revision A, notification *note*, view $\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$
Output: refined view
1: **if** $note = true$ **then return** $\langle \mathcal{B}^+, \ \mathcal{B}^- \cup \{(\mathsf{con}(\mathcal{C}) \wedge A, \mathit{false})\}, \ \mathcal{C} \cup \{A\} \rangle$
2: **else return** $\langle \mathcal{B}^+ \cup \{(\mathsf{con}(\mathcal{C}) \wedge A, \mathit{false})\}, \ \mathcal{B}^-, \ \mathcal{C} \rangle$

can rule out all consequence relations by which $\mathcal{D}$ would reason differently from the already observed reasoning behavior (described in $\mathcal{B}^+$ and $\mathcal{B}^-$). Finally, $\mathcal{A}$ is aware that $\mathcal{D}$ always considers her current assertions possible so that $\mathcal{A}$ rules out all consequence relations that map $\mathcal{C}$ to *false* (Prop. 1). Then, $\mathcal{A}$ determines which beliefs $\mathcal{D}$ can derive in all the remaining, possible consequence relations from (the visible part $\mathcal{C}$ of) her current assertions.

Definition 7 (Possible Consequence Relation). *Let $\mathcal{V} = \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ be a view. Then, a consequence relation $\mathrel{|\!\sim}' \subseteq \mathcal{L}_{pl} \times \mathcal{L}_{pl}$ is possible under $\mathcal{V}$ iff the following conditions apply:*
(Defined by OCF) There exists an OCF κ such that $\mathrel{|\!\sim}_\kappa = \mathrel{|\!\sim}'$.
($\mathcal{B}$ Compatible) $\mathcal{B}^+ \subseteq \mathrel{|\!\sim}'$ and $\mathcal{B}^- \subseteq \not\mathrel{|\!\sim}'$.
($\mathcal{C}$ Consistent) $\mathsf{con}(\mathcal{C}) \not\mathrel{|\!\sim}'$ false.

Definition 8 (Skeptical Entailment). *Let $\mathcal{V} = \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ be a view. Then, skeptical entailment from $\mathcal{V}$ is defined by*

$$\mathsf{skeptical}(\mathcal{V}) = \{B \in \mathcal{L}_{pl} \mid \ \textit{for each} \mathrel{|\!\sim}' \textit{possible under } \mathcal{V}\textit{:} \ \mathsf{con}(\mathcal{C}) \mathrel{|\!\sim}' B\}.$$

4.2 Skeptical Reasoning with Hidden Assertions

Usually, $\mathcal{D}$ may have acquired parts of her current assertions from other sources than agent $\mathcal{A}$, e.g., other agents or sensors, before interacting with $\mathcal{A}$. Moreover, $\mathcal{D}$ may not want to reveal all of her current assertions to $\mathcal{A}$ to conceal some confidential beliefs (cf. Prop. 2: $\mathcal{D}$'s current assertions are part of her deductively closed belief). Thus in the remainder of the article, we drop the assumption, that $\mathcal{R}$ is visible to $\mathcal{A}$, and instead consider that $\mathcal{D}$ intentionally leaves $\mathcal{A}$ uncertain about some of her initial assertions during their interaction. This means that agent $\mathcal{A}$ considers possible that other sources contributed further assertions $\mathcal{E} \subset_{fin} \mathcal{L}_{pl}$ to the visible part $\mathcal{C}$ of $\mathcal{D}$'s current assertions before the interaction:

Definition 9 (Possible Extension). *Let $\mathcal{V} = \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ be a view and $\mathcal{E} \subset_{fin} \mathcal{L}_{pl}$ a finite set of assertions. Then, $\mathcal{E}$ is a possible extension (of $\mathcal{C}$) under $\mathcal{V}$ iff there exists a consequence relation $\mathrel{|\!\sim}'$ possible under $\mathcal{V}$ s.t. $\mathsf{con}(\mathcal{C} \cup \mathcal{E}) \not\mathrel{|\!\sim}'$ false.*

oarieli@mta.ac.il

In this situation, under her view $\mathcal{V}_0$ agent $\mathcal{A}$ may choose some possible extension $\mathcal{E}$, assume that actually $\mathcal{R} = \mathcal{C}_0 \cup \mathcal{E}$ initially and, under this assumption, compute her approximations during an uncontrolled interaction with $\mathcal{D}$ as if that $\mathcal{R}$ was visible. In this way, $\mathcal{A}$ can employ Proc. 1 and Proc. 2 for her approximations. We denote the computed view by $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H})$ referring to the initial view $\mathcal{V}_0$, to the chosen extension $\mathcal{E}$ and to the history $\mathcal{H}$ of the interaction.

Example 5. Consider the initial view $\mathcal{V}_0 = \langle \emptyset, \emptyset, \emptyset \rangle$. Here, $\mathcal{A}$ may simply assume the extension $\mathcal{E} = \{s_3\}$. Then, during the interaction of Ex. 3 she can do the same approximations as in Ex. 4.

As a skeptical reasoner, $\mathcal{A}$ is interested in all beliefs that she can skeptically conclude independently from the possible extension that she has chosen initially. For a feasible computation of these beliefs the following proposition is essential in showing that such a computation only needs to consider the empty extension $\mathcal{E} = \emptyset$. This property is exploited by the procedures in Section 5.2 for the computation of $\mathcal{A}$'s approximations and her skeptical entailment. For simplifying the proofs considerably, for the following result and the remainder of this article we make the following assumption:

Assumption 5. *Agent $\mathcal{A}$ is initially ignorant about $\mathcal{D}$'s fixed reasoning: $\mathcal{B}_0 = \emptyset$.*

At the end of this section, we briefly discuss the general case.

Proposition 3 (Skeptical Reasoning with Hidden Assertions). *Let $\mathcal{V}_0 = \langle \emptyset, \emptyset, \mathcal{C}_0 \rangle$ be the initial view and $\mathcal{H}$ a history. Then,*

1. *$\mathcal{E} \subset_{fin} \mathcal{L}_{pl}$ is a possible extension under $\mathcal{V}_0$ iff $\mathcal{C}_0 \cup \mathcal{E}$ is satisfiable;*

2. $\mathsf{skeptical}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H})) = \{B \in \mathcal{L}_{pl} \mid B \in \mathsf{skeptical}(\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H}))$ *for all possible extensions $\mathcal{E} \subset_{fin} \mathcal{L}_{pl}$ under $\mathcal{V}_0$*$\}$.

Sketch of Proof. Proposition 3.2 follows from the fact that agent $\mathcal{A}$ has the following two equivalent options to find a consequence relation that explains $\mathcal{D}$'s reactions in an uncontrolled interaction (if $\mathcal{V}_0 = \langle \emptyset, \emptyset, \mathcal{C}_0 \rangle$):[4]

(1) assume the visible part $\mathcal{C}_0$ as $\mathcal{D}$'s initial assertions, and after the interaction with history $\mathcal{H}$, find a possible consequence relation under $\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H})$;
(2) choose a possible extension $\mathcal{E}$ under $\mathcal{V}_0$, assume $\mathcal{C}_0 \cup \mathcal{E}$ as $\mathcal{D}$'s initial assertions, and find a possible consequence relation under $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H})$.

In summary, if $\mathcal{A}$ does not skeptically entail B, using the second option (i.e., B is not in the set on the right-hand side in Prop. 3. 2), she would neither skeptically entail B, using the first option (i.e., $B \notin \mathsf{skeptical}(\mathcal{V}(\mathcal{V}_0, \emptyset, \mathcal{H}))$. The equivalence of the two options is subject of the following lemma. For applying this lemma, we notice that all conclusions represented in $\mathcal{B}(\mathcal{V}_0, \emptyset, \mathcal{H})$, on the one hand, and $\mathcal{B}(\mathcal{V}_0, \mathcal{E}, \mathcal{H})$, on the other hand, differ only in their premises. In the latter case, all premises are extended by $\mathsf{con}(\mathcal{E})$ so that $\mathcal{V}(\mathcal{V}_0, \mathcal{E}, \mathcal{H})$ fulfills the two properties required by the lemma. □

[4] The inclusion "$\supseteq$" in Prop. 3. 2 holds by definition.

oarieli@mta.ac.il

Lemma 1 (Skeptical Reasoning with Hidden Assertions). *Let $\mathcal{V} = \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$ be a view (with no further restrictions). Then, there exists a possible consequence relation under $\mathcal{V}$ iff there exist a set $\mathcal{E} \subset_{fin} \mathcal{L}_{pl}$ (not necessarily a possible extension) and a possible consequence relation under $\mathcal{V}_\mathcal{E} = \langle \mathcal{B}^+_\mathcal{E}, \mathcal{B}^-_\mathcal{E}, \mathcal{C}_\mathcal{E} \rangle$ where*

1. *for all $P, C \in \mathcal{L}_{pl}$ and $s \in \{+, -\}$: $(P, C) \in \mathcal{B}^s$ iff $(P \wedge \mathsf{con}(\mathcal{E}), C) \in \mathcal{B}^s_\mathcal{E}$,*
2. *$\mathcal{C}_\mathcal{E} = \mathcal{C} \cup \mathcal{E}$.*

Sketch of Proof. The only-if-part obviously holds with $\mathcal{E} = \emptyset$. For the if-part, we assume a set $\mathcal{E} \subset_{fin} \mathcal{L}_{pl}$ such that there is a possible consequence relation $\mathrel{|\!\sim}'_\mathcal{E}$ under $\mathcal{V}_\mathcal{E}$ defined by an OCF $\kappa'_\mathcal{E}$. Then, we define a possible consequence relation under $\mathcal{V}$ by an OCF κ' as follows: The idea is that κ' defines the plausibility of models of $\mathsf{con}(\mathcal{E})$ by $\kappa'_\mathcal{E}$ whereas it makes models of $\neg\mathsf{con}(\mathcal{E})$ impossible:

$$\kappa'(\omega) := \begin{cases} \kappa'_\mathcal{E}(\omega) & \text{if } \omega \models_{pl} \mathsf{con}(\mathcal{E}), \\ \infty & \text{otherwise.} \end{cases}$$

In words, κ' lifts up all worlds to rank ∞ that are not relevant for conclusions in $\mathcal{B}_\mathcal{E}$ (i.e., their premise is not satisfied) but possibly not for conclusions in $\mathcal{B}$ (i.e., their premise may be satisfied). The key to proving that κ' defines a possible consequence relation under $\mathcal{V}$ is showing that for each $P \in \mathcal{L}_{pl}$ the most plausible models of P under κ' are exactly all most plausible models of $P \wedge \mathsf{con}(\mathcal{E})$ under $\kappa'_\mathcal{E}$. Afterwards, we use Def. 1, the two properties of $\mathcal{V}_\mathcal{E}$ and the assumption that $\mathrel{|\!\sim}'_\mathcal{E}$ is possible under $\mathcal{V}_\mathcal{E}$ to finish the proof. □

In future work, we may consider the general situation that $\mathcal{B}_0 \neq \emptyset$ (i.e., dropping Assumption 5). However, if agent $\mathcal{D}$ has hidden parts of her assertions from agent $\mathcal{A}$ before the interaction with $\mathcal{A}$, we have to clarify which parts of $\mathcal{B}_0$ are actually useful for agent $\mathcal{A}$. For example, consider the initial view $\mathcal{V}_0 = \langle (s_1, c), \emptyset, \{s_1\} \rangle$ which says that $\mathcal{A}$ is aware that $\mathcal{D}$ by her medical expertise will defeasibly conclude cancer from symptom s_1 and that $\mathcal{D}$ asserts that the patient shows symptom s_1. Yet, having gathered further (hidden) assertions, agent $\mathcal{D}$ might already have withdrawn this conclusion (cf. Ex. 2 where $s_1 \mathrel{|\!\sim}_{\kappa_1} c$, but $s_1 \wedge s_4 \mathrel{|\!\sim}_{\kappa_1} \neg c$ (see l_{11}, l_{12} and l_{13})).

5 A Confidentiality Preserving Agent

5.1 Requirements and Architecture

In Sect. 1, we have already outlined the additional requirements on agent $\mathcal{D}$ under the confidentiality aspect: $\mathcal{D}$ specifies confidential beliefs in a policy $conf \subset_{fin} \mathcal{L}_{pl}$. Here $B \in conf$ means that agent $\mathcal{A}$ should not be able to skeptically conclude that $\mathcal{D}$ *currently* believes B. Note that previous beliefs already given up may be revealed to $\mathcal{A}$. Following Ex. 3, with $conf = \{c\}$ the situation $c \in \mathsf{Bel}(\langle \kappa_1, \{s_3, s_1 \vee s_2\} \rangle)$ must be hidden from $\mathcal{A}$. After $\mathcal{D}$ revised her belief base $\langle \kappa_1, \{s_3, s_1 \vee s_2\} \rangle$

oarieli@mta.ac.il

with $\mathcal{A}$'s information s_4, the dismissed diagnosis of cancer can be revealed to $\mathcal{A}$. The semantics of $\mathcal{D}$'s confidentiality policy is formalized by Def. 10 below. For enforcing confidentiality, Biskup et al in [6] proposed to augment a BDI architecture with three additional components, the view $\mathcal{V}$, the policy *conf*, and a control component `cexec`, the *censor*. We illustrate this architecture in Fig. 2 with the censor for query requests that we will detail in Sect. 5.2. This censor may refuse requests to preserve confidentiality in contrast to the *lying* approach [4] where misinformation would be given.

In the view $\mathcal{V}$, agent $\mathcal{D}$ keeps track of $\mathcal{A}$'s approximations $\mathcal{B}^+$, $\mathcal{B}^-$ and $\mathcal{C}$. We assume initially (Assumption 1) and ensure during the interaction that the view maintained by $\mathcal{D}$ equals $\mathcal{A}$'s actual view that has been defined in Sect. 4. In particular, we can guarantee confidentiality only under the above assumption that $\mathcal{D}$ acquired $\mathcal{A}$'s initial approximations. Recall that, moreover, Assumption 5 allows only $\mathcal{B}_0 = \emptyset$. Furthermore, to guarantee confidentiality, agent $\mathcal{D}$ must ensure before interacting with $\mathcal{A}$ that the confidentiality policy is not breached initially and that $\mathcal{D}$'s current assertions are initially approximated correctly. These initial preconditions are summarized by the following predicate:

$$\texttt{pre}(\langle\kappa,\mathcal{R}\rangle, \mathit{conf}, \mathcal{V}_0): \quad \mathcal{B}_0^+ = \mathcal{B}_0^- = \emptyset \text{ and } \mathcal{C}_0 \not\vdash_{pl} B \text{ for all } B \in \mathit{conf} \text{ and } \mathcal{R} \vdash_{pl} F \text{ for all } F \in \mathcal{C}_0. \tag{1}$$

After ensuring these preconditions, during an interaction with agent $\mathcal{A}$ (Sect. 3) agent $\mathcal{D}$ employs the censor `cexec` aiming at guaranteeing confidentiality as outlined in Sect. 1 and formalized as follows:

Definition 10 (Confidentiality Preservation). *Agent $\mathcal{D}$ preserves confidentiality (of belief) towards agent $\mathcal{A}$ iff*

for all finite sequences I of requests,
confidentiality policies conf, initial views $\mathcal{V}_0$, belief bases $\langle\kappa,\mathcal{R}\rangle$ such that $\langle\kappa,\mathcal{R}\rangle$, conf and $\mathcal{V}_0$ satisfy `pre`*,*
for all $B \in \mathit{conf}$ there exists an alternative belief base $\langle\kappa',\mathcal{R}'\rangle$ such that $\langle\kappa',\mathcal{R}'\rangle$, conf and $\mathcal{V}_0$ satisfy `pre` *and*

1. $\texttt{cexec}(\langle\kappa,\mathcal{R}\rangle, \mathit{conf}, \mathcal{V}_0, I) = \texttt{cexec}(\langle\kappa',\mathcal{R}'\rangle, \mathit{conf}, \mathcal{V}_0, I)$*, and*
2. $B \notin \mathsf{Bel}(\langle\kappa'_I,\mathcal{R}'_I\rangle)$ *where $\langle\kappa'_I,\mathcal{R}'_I\rangle$ is $\mathcal{D}$'s belief base after $\mathcal{D}$ has reacted to the sequence I starting with $\langle\kappa',\mathcal{R}'\rangle$.*

5.2 A Confidentiality Preserving Censor

As known in the area of classical database systems [4,8], a refusal-based censor must deal with *meta-inference*, i.e., a database user's reasoning about the processing of the censor. Since the user knows the procedure of the censor and sees all input but the database (cf. Assumption 3), he can determine all answers that the censor is going to refuse. In the worst case, the censor is going to refuse only one answer so that the refusal reveals this answer. The common strategy against meta-inferences is to refuse additional answers so that tracing back presumably correct answers from a refusal will not leak secrets.

oarieli@mta.ac.il

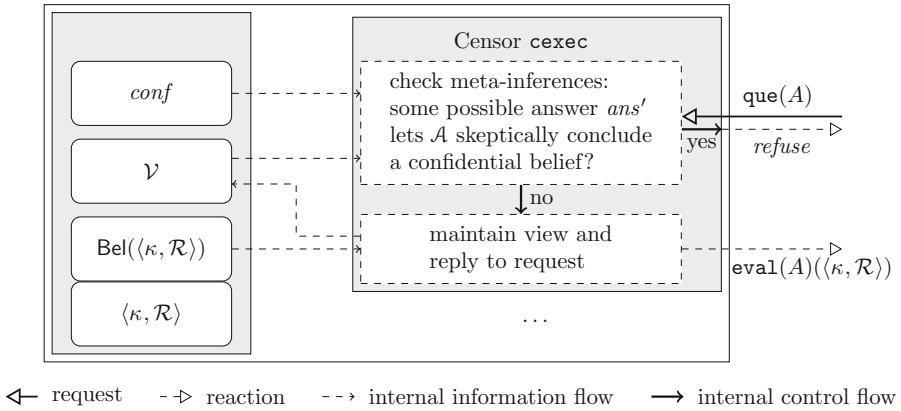


Fig. 2. Controlled reactions of agent $\mathcal{D}$ to a query request

For controlled query-answering, we equip agent $\mathcal{D}$ with the censor illustrated by Fig. 2 and implemented by Proc. 3:

1. First (line 1-2), the censor determines whether a value $ans \in \{true, false, undef\}$ is a *possible answer* to the query request $\texttt{que}(A)$, i.e., an answer that agent $\mathcal{A}$ does not rule out given her approximations $\mathcal{V}$:

   The censor refines $\mathcal{V}$ by the answer *ans* (app_by_que, Sect. 4). Then, the censor checks, whether the refinement still can approximate $\mathcal{D}$'s fixed reasoning, by evaluating the following predicate on the refined view:

$$\texttt{poss}(\mathcal{V}): \text{ there exists } \sim' \text{ possible under } \mathcal{V}. \tag{2}$$

2. Next (line 1-2), the censor checks whether returning a possible answer *ans* enables agent $\mathcal{A}$ to skeptically conclude a confidential belief.
3. In case of such a conclusion (line 3), the censor refuses the query.
4. Otherwise (line 6-7), the censor maintains the view, evaluates the query and sends the correct answer to $\mathcal{A}$.

Proc. 3 is adopted from [4] but here the answer reveals belief or non-belief, and the derivation or non-derivation of belief from agent $\mathcal{D}$'s current assertions is described by subsets of $\mathcal{L}_{pl} \times \mathcal{L}_{pl}$ (Sect. 4). Meta-inference is prevented as shown by Theorem 1 below. The reason is that a query is refused depending on $\mathcal{V}$ and *conf* only (line 1-5), but not on the belief base (used in line 6-7). Note that Proc. 3 does not refine the view after a refusal since by the above considerations about meta-inference a refusal does not provide information about $\mathcal{D}$'s belief base. Therefore, $\mathcal{A}$'s approximations and skeptical entailment based thereupon, simulated by the censor during a controlled interaction, do not differ from those during uncontrolled interactions in Sect. 4.

Example 6. To illustrate the censor for controlled query-answering, we consider the policy $conf = \{c\}$ and $\mathcal{D}$'s initial belief base $\langle\kappa,\mathcal{R}\rangle = \langle\kappa_1, \{s_3, s_1 \vee s_2\}\rangle$

oarieli@mta.ac.il

Procedure 3. `cexec` for controlled query execution

Input: request $\mathtt{que}(A)$, view $\mathcal{V}$, policy *conf*, belief base $\langle\kappa,\mathcal{R}\rangle$

1: **for** $ans \in \{true, false, undef\}$ **do**
2: **if** $\mathtt{poss}(\mathtt{app_by_que}(A, ans, \mathcal{V}))$ **and** there exists $B \in \mathsf{skeptical}(\mathtt{app_by_que}(A, ans, \mathcal{V}))$[5] with $B \in conf$ **then**
3: Notify $\mathcal{A}$: *refuse*. `exit`
4: **end if**
5: **end for**
6: $\mathcal{V} := \mathtt{app_by_que}(A, \mathtt{eval}(A)(\langle\kappa,\mathcal{R}\rangle), \mathcal{V})$
7: Notify $\mathcal{A}$: $\mathtt{eval}(A)(\langle\kappa,\mathcal{R}\rangle)$

(cf. Ex. 2) on which $\mathcal{A}$ has the view $\mathcal{V}_0 = \langle\emptyset, \{s_1 \vee s_2\}\rangle$. This initial situation complies with the preconditions of the censor in (1). Now, agent $\mathcal{A}$ initiates the query request $\mathtt{que}((s_1 \wedge c) \vee s_2)$.

Next we demonstrate the censor's check of whether the answer *false* is a possible answer and reveals the confidential belief $c \in conf$. The values *true* and *undef* can be treated by similar arguments. The answer *false* is possible since there is a possible consequence relation $\sim'$ under the refined view $\mathcal{V}' = \langle\{(s_1 \vee s_2, \neg((s_1 \wedge c) \vee s_2))\}, \emptyset, \{s_1 \vee s_2\}\rangle$ (recall that $s_1 \vee s_2 \sim' \neg((s_1 \wedge c) \vee s_2)$ and $s_1 \vee s_2 \not\sim' false$ must hold (Def. 7)). This consequence relation is defined by the OCF κ'

1	s_1	$*$	$*$	$*$	c	s_1	s_2	$*$	$*$	$\neg c$	$\neg s_1$ $*$ $*$ $*$ $*$
0	s_1	$\neg s_2$	$*$	$*$	$\neg c$.						

Moreover, the consequence relation $\sim'$ verifies that the answer *false* does not enable agent $\mathcal{A}$ to skeptically conclude that $\mathcal{D}$ believes c. The reason is that $\sim'$ is possible under $\mathcal{V}'$ and $s_1 \vee s_2 \not\sim' c$. After a check of values *true* and *undef* with a similar result, the actual answer *true* is returned to $\mathcal{A}$ and the view updated to $\mathcal{V}_1 = \langle\{(s_1 \vee s_2, (s_1 \wedge c) \vee s_2)\}, \emptyset, \{s_1 \vee s_2\}\rangle$.

In a controlled revision processing, if it is successful, agent $\mathcal{D}$ adds information A received from agent $\mathcal{A}$ by the request $\mathtt{rev}(A)$ to her current assertions. Here, a notification of success/failure implicitly informs $\mathcal{A}$ whether $\mathcal{D}$ considers the assertions $\mathcal{R} \cup \{A\}$ possible or impossible (Sect. 3). Sometimes the notification must be suppressed if it enables $\mathcal{A}$ to skeptically conclude some confidential belief. But checking whether such a suppression is needed must be independent from $\mathcal{D}$ considering $\mathcal{R} \cup \{A\}$ possible or not. Otherwise, suppressing a notification may leak information about the belief base. Such a leakage would be caused by agent $\mathcal{A}$'s meta-inference (about the suppression).

The censor implemented by Procedure 4 proceeds in three steps:

1. First (line 1-3), the censor checks whether after a successful revision agent $\mathcal{A}$ is able to skeptically conclude a confidential belief $B \in conf$. In such a case, the revision is refused.
2. Otherwise (line 4-10), the censor checks whether a success/failure notification breaches confidentiality. There are the following three cases:

[5] By Prop. 3 the censor can simplify the computation of skeptical entailment this way.

oarieli@mta.ac.il

Procedure 4. `cexec` for controlled revision execution

Input: request $\texttt{rev}(A)$, view $\mathcal{V} = \langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \rangle$, policy *conf*, belief base $\langle \kappa, \mathcal{R} \rangle$

1: **if** there exists $B \in \mathsf{skeptical}(\texttt{app_by_rev}(A, \mathit{true}, \mathcal{V}))$ with $B \in \mathit{conf}$ **then**
2: Notify $\mathcal{A}$: "Revision would breach confidentiality". `exit`
3: **end if**
4: **if** $\texttt{poss}(\texttt{app_by_rev}(A, \mathit{false}, \mathcal{V}))$ **then**
5: **if** $\mathit{false} \in \mathsf{skeptical}(\langle \mathcal{B}^+, \mathcal{B}^-, \mathcal{C} \cup \{A\} \rangle)$ **then**
6: Notify $\mathcal{A}$: *false*. `exit`
7: **else if** there exists $B \in \mathsf{skeptical}(\texttt{app_by_rev}(A, \mathit{false}, \mathcal{V}))$ with $B \in \mathit{conf}$ **then**
8: Notify $\mathcal{A}$: "Failure notification would breach confidentiality". `exit`
9: **end if**
10: **end if**
//Process revision
11: **if** $\texttt{notify}(A)(\langle \kappa, \mathcal{R} \rangle) = \mathit{true}$ **then**
12: $\mathcal{R} := \mathcal{R} \cup \{A\}$
13: **end if**
14: $\mathcal{V} := \texttt{app_by_rev}(A, \texttt{notify}(A)(\langle \kappa, \mathcal{R} \rangle), \mathcal{V})$
15: Notify $\mathcal{A}$: $\texttt{notify}(A)(\langle \kappa, \mathcal{R} \rangle)$.

(1) $\mathcal{A}$ knows whether $\mathcal{D}$ considers the visible assertions $\mathcal{C}$ together with assertions A possible (line 4) or not (line 5-6).
(2) Revealing that these assertions are considered impossible through a failure notification discloses a confidential belief (line 7-9).
(3) Revealing that these assertions are considered possible through a success notification discloses a confidential belief (line 1-3).
In case (1), the censor processes the revision either successfully (line 11-15, known possibility result) or with a failure (line 6, known impossibility result). In case (2), the revision is refused. (Case (3) is captured by line 1-3.)

3. Otherwise (line 11-15), the censor processes the revision, maintains the view and sends a success/failure notification to $\mathcal{A}$.

Example 7. We continue Example 6 to illustrate the censor for controlled revision processing. Processing agent $\mathcal{A}$'s request $\texttt{rev}(s_2)$, the censor passes until line 6 (we omit the explanations here) and in line 7 detects that a failure notification would enable agent $\mathcal{A}$ to skeptically conclude that $\mathcal{D}$ believes c:

$$c \in \mathsf{skeptical}(\mathcal{V}') \text{ where } \mathcal{V}' = \texttt{app_by_rev}(s_2, \mathit{false}, \mathcal{V}_1) = \langle \{(s_1 \vee s_2, (s_1 \wedge c) \vee s_2), ((s_1 \vee s_2) \wedge s_2, \mathit{false})\}, \emptyset, \{s_1 \vee s_2\} \rangle. \quad (3)$$

Next we justify (3) by elementary arguments (we discuss a computation in Sect. 6). Equation (3) says that for any possible consequence relation $\mid\!\sim'$ under $\mathcal{V}'$ it holds that $s_1 \vee s_2 \mid\!\sim' c$. Thus, consider a possible consequence relation $\mid\!\sim'$ under $\mathcal{V}'$ defined by an OCF κ':

1. By Def. 7, we have $(s_1 \vee s_2) \wedge s_2 \mid\!\sim' \mathit{false}$ and $s_1 \vee s_2 \not\mid\!\sim' \mathit{false}$. Because $\mid\!\sim'$ is defined on the basis of models of formulas (Def. 2) and $(s_1 \vee s_2) \wedge s_2$ is

oarieli@mta.ac.il

equivalent to s_2, it follows that $s_2 \mathrel{\sim'}$ *false* holds. Altogether, by Prop. 1 it holds that $\kappa'(s_2) = \infty$ while $\kappa'(s_1 \vee s_2) \neq \infty$.

2. To prove $s_1 \vee s_2 \mathrel{\sim'} c$, we show $\omega \models_{pl} c$ for any most plausible model ω of $s_1 \vee s_2$, i.e., a model such that $\kappa'(\omega) = \kappa'(s_1 \vee s_2) \neq \infty$ (Def. 2). By Def. 7, the possible consequence relation $\mathrel{\sim'}$ satisfies $s_1 \vee s_2 \mathrel{\sim'} (s_1 \wedge c) \vee s_2$. Thus, it holds that $\omega \models_{pl} (s_1 \wedge c) \vee s_2$ and, moreover, $\omega \models_{pl} \neg s_2$ because $\kappa'(s_2) = \infty \neq \kappa'(s_1 \vee s_2) = \kappa'(\omega)$. Therefore, $\omega \models_{pl} c$ must hold.

The censor exits execution without a revision (line 8) and notifies $\mathcal{A}$ accordingly. The notification is specific for line 8 and therefore reveals to agent $\mathcal{A}$ that the censor exits in this line and, thus, without a revision. Apart from this, the notification has no further informational value since, by Assumption 3, agent $\mathcal{A}$ can freely access all arguments used in the computations up to line 8.

Theorem 1 (Confidentiality Preservation). *Agent $\mathcal{D}$ preserves confidentiality of belief towards $\mathcal{A}$ according to Def. 10 by the means of the two procedures of* `cexec`.

Sketch of Proof. Starting with any initial situation of agent $\mathcal{D}$, after any finite sequence I of requests by agent $\mathcal{A}$, we construct an alternative initial belief base $\langle \kappa'_B, \mathcal{R}'_B \rangle$ for each confidential belief $B \in \mathit{conf}$ as follows.

First, we can prove that the censor `cexec` preserves the following invariant by induction on the length k of I:

$$B \notin \mathsf{skeptical}(\mathcal{V}^k) \text{ for each } B \in \mathit{conf}, \tag{4}$$

where $\mathcal{V}^k$ denotes the view computed by the censor in the k-th request. The main argument is that the precondition (1) and both procedures explicitly ensure the invariant but the latter depending on the validity of the predicate $\mathtt{poss}(\mathcal{V}^k)$. We will exemplify this argument for the censor for controlled query execution with a request $\mathtt{que}(A)$. Previously, the view might have been modified only by either a correct answer in case of a previous query request (see line 6) or a correct notification in case of a previous revision request (see line 14). Therefore, we intuitively see that $\mathcal{D}$'s fixed reasoning $\mathrel{\sim_\kappa}$ is possible under the view $\mathcal{V}^k_{\mathcal{R}_0}$ defined in Lemma 1 (corresponding to agent $\mathcal{A}$'s approximations when $\mathcal{D}$'s initial assertions $\mathcal{R}_0$ were visible). Consequently, by Lemma 1, there is a possible consequence relation under $\mathcal{V}^k$ so that the predicate $\mathtt{poss}(\mathcal{V}^k)$ holds. Finally, line 2 enforces the invariant (4) with $\mathcal{V}^k = \mathtt{app_by_que}(A, \mathtt{eval}(A)(\langle \kappa, \mathcal{R} \rangle), \mathcal{V}^{k-1})$.

The invariant guarantees the existence of a possible consequence relation $\mathrel{\sim_{\kappa'_B}}$ under the final view $\mathcal{V}_I$ after the interaction such that

- $\mathcal{D}$ does not defeasibly conclude B from the visible assertions $\mathcal{C}_I$ after the interaction: $\mathsf{con}(\mathcal{C}_I) \not\mathrel{\sim_{\kappa'_B}} B$;
- in this alternative situation after the interaction, $\mathcal{D}$ does not believe B: $B \notin \mathsf{Bel}(\langle \kappa'_B, \mathcal{C}_I \rangle)$.

Finally, we can prove that agent $\mathcal{A}$ cannot distinguish the alternative and $\mathcal{D}$'s actual initial belief base after the interaction by $\mathcal{D}$'s reactions. The main argument is that $\mathcal{D}$'s reactions with the alternative belief base are entirely determined by the view $\mathcal{V}_I$ that the censor has computed according to $\mathcal{D}$'s reactions. □

oarieli@mta.ac.il

6 Summary and Discussion

We presented a censor that provably enforces the confidentiality interests of an agent $\mathcal{D}$ when she reacts to revisions and queries about her current belief from another agent $\mathcal{A}$. The crucial task of the censor is to follow $\mathcal{A}$'s skeptical entailment of $\mathcal{D}$'s belief while $\mathcal{D}$ has hidden some part of her current assertions from $\mathcal{A}$. We have shown that the censor needs only consider skeptical entailment based on the visible assertions, when $\mathcal{A}$ is initially ignorant about $\mathcal{D}$'s fixed reasoning, and briefly discussed the general situation. For implementing $\mathcal{D}$'s reasoning component, we may apply complexity results [11] and algorithms [13] tailored to related approaches [12,2] of defeasible reasoning. Further, Agent $\mathcal{D}$ can decide whether $\mathcal{A}$ skeptically entails $B \in \mathcal{L}_{pl}$ by a satisfiability test in propositional conditional logic, cf. [12].

Our work is closely related to the work of Booth and Nittka [9]. However, in their proposal, $\mathcal{A}$ gives a "rational explanation" for her observations of $\mathcal{D}$'s reactions that is different from skeptical entailment. More significantly, the revision process in [9] is a kind of merging of all assertions that $\mathcal{D}$ gathered so far giving precedence to new information if inconsistency must be repaired. For example, in the situation of Ex. 7 agent $\mathcal{D}$ may merge the list of her assertions $\langle s_3, s_1 \vee s_2, s_2 \rangle$ ordered by recency respecting her unquestionable belief of $\neg s_2 \vee \neg s_3$, cf. Sect. 3. The merging yields $(s_1 \vee s_2) \wedge s_2 \wedge (\neg s_2 \vee \neg s_3) =: A$ with the operator in [9]. Now in [9], $\mathcal{D}$'s belief are all propositional consequences of the formula A, i.e., $\mathsf{Bel}(\vdash_{pl}, A)$ in terms of our definitions. Altogether, because of these differences, we had to adapt their approach to our scenario and to an observer $\mathcal{A}$ as a skeptical reasoner.

In future work, we may augment $\mathcal{D}$'s belief $\mathsf{Bel}(\langle \kappa, \mathcal{R} \rangle)$ with degrees of beliefs where a higher degree expresses more confidence in a belief [1]. Degrees of belief can be used to specify a finer grained confidentiality policy. Wiese in [17] uses such policies for query-answering to a possibilistic knowledge base. Confidentiality is preserved by weakening answers which "is a more cooperative way of communication than denying access altogether".

As outlined in Sect. 1, there is the special case of $\mathcal{D}$'s fixed reasoning being propositional entailment $\vdash_{pl}$ so that agent $\mathcal{D}$'s belief $\mathsf{Bel}(\vdash_{pl}, \mathcal{R})$ consists of all formulas propositionally entailed by the propositional incomplete database $\mathcal{R}$. Such databases are studied in [8] for confidentiality preserving query-answering only, but there the censor is not as restrictive as Procedure 3. For example, if only returning *true* to query $\mathsf{que}(A)$ was harmful, it would suffice to refuse the answer *true* and, say, *false*. The answer *undef* could be returned safely in contrast to our approach. In this example, a refusal reveals to agent $\mathcal{A}$ that agent $\mathcal{D}$ believes A or believes $\neg A$. This fact can be expressed in propositional modal logic in the special case of [8]. In general ($\sim \neq \vdash_{pl}$), the fact that $\mathcal{D}$ defeasibly concludes A from $\mathcal{R}$ or defeasibly concludes $\neg A$ from $\mathcal{R}$ cannot be expressed using the relations $\mathcal{B}^+$ and $\mathcal{B}^-$: $(\mathsf{con}(\mathcal{C}), A)$ is neither definitely in $\mathcal{B}^+$ nor in $\mathcal{B}^-$. Instead, this fact may be represented by the formula $(\mathsf{con}(\mathcal{C}) \rightarrow A) \vee (\mathsf{con}(\mathcal{C}) \rightarrow \neg A)$ in propositional conditional logic (where $\rightarrow$ denotes the conditional operator) so that Procedure 3 may be enhanced following [8].

oarieli@mta.ac.il

References

1. Beierle, C., Kern-Isberner, G.: A Conceptual Agent Model Based on a Uniform Approach to Various Belief Operations. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 273–280. Springer, Heidelberg (2009)
2. Benferhat, S., Dubois, D., Prade, H.: Nonmonotonic reasoning, conditional objects and possibility theory. Artif. Intell. 92(1-2), 259–276 (1997)
3. Biskup, J.: Usability Confinement of Server Reactions: Maintaining Inference-Proof Client Views by Controlled Interaction Execution. In: Kikuchi, S., Sachdeva, S., Bhalla, S. (eds.) DNIS 2010. LNCS, vol. 5999, pp. 80–106. Springer, Heidelberg (2010)
4. Biskup, J., Bonatti, P.A.: Controlled query evaluation for enforcing confidentiality in complete information systems. Int. J. Inf. Sec. 3(1), 14–27 (2004)
5. Biskup, J., Gogolin, C., Seiler, J., Weibert, T.: Inference-proof view update transactions with forwarded refreshments. Journal of Computer Security 19(3), 487–529 (2011)
6. Biskup, J., Kern-Isberner, G., Thimm, M.: Towards enforcement of confidentiality in agent interactions. In: Pagnucco, M., Thielscher, M. (eds.) NMR 2008. pp. 104–112. The University of New South Wales (2008)
7. Biskup, J., Tadros, C.: Inference-Proof View Update Transactions with Minimal Refusals. In: Garcia-Alfaro, J. (ed.) DPM 2011 and SETOP 2011. LNCS, vol. 7122, pp. 104–121. Springer, Heidelberg (2012)
8. Biskup, J., Weibert, T.: Keeping secrets in incomplete databases. Int. J. Inf. Sec. 7(3), 199–217 (2008)
9. Booth, R., Nittka, A.: Reconstructing an agent's epistemic state from observations about its beliefs and non-beliefs. J. Log. Comput. 18(5), 755–782 (2008)
10. Dubois, D.: Three scenarios for the revision of epistemic states. J. Log. Comput. 18(5), 721–738 (2008)
11. Eiter, T., Lukasiewicz, T.: Complexity results for default reasoning from conditional knowledge bases. In: Cohn, A.G., Giunchiglia, F., Selman, B. (eds.) KR 2000, pp. 62–73. Morgan Kaufmann, San Francisco (2000)
12. Friedman, N., Halpern, J.Y.: Plausibility measures and default reasoning. J. ACM 48(4), 648–685 (2001)
13. Garcia, B.B., Brasil Jr., S.M.: Towards Default Reasoning through MAX-SAT. In: Bittencourt, G., Ramalho, G.L. (eds.) SBIA 2002. LNCS (LNAI), vol. 2507, pp. 52–62. Springer, Heidelberg (2002)
14. Goldszmidt, M., Pearl, J.: On the consistency of defeasible databases. Artif. Intell. 52(2), 121–149 (1992)
15. Kern-Isberner, G.: Linking iterated belief change operations to nonmonotonic reasoning. In: Brewka, G., Lang, J. (eds.) KR 2008, pp. 166–176. AAAI Press, Menlo Park (2008)
16. Spohn, W.: Ordinal conditional functions: A dynamic theory of epistemic states. In: Skyrms, B., Harper, W.L. (eds.) Irvine Conference on Probability and Causation. Causation in Decision, Belief Change, and Statistics, vol. II, pp. 105–134. Kluwer, Dordrecht (1988)
17. Wiese, L.: Keeping Secrets in Possibilistic Knowledge Bases with Necessity-Valued Privacy Policies. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IPMU 2010. LNCS, vol. 6178, pp. 655–664. Springer, Heidelberg (2010)
18. Wooldridge, M.J.: An Introduction to Multiagent Systems. Wiley, Chichester (2009)

oarieli@mta.ac.il

More Than the Sum of Its Parts – Holistic Ontology Alignment by Population-Based Optimisation

Jürgen Bock[1], Sebastian Rudolph[2], and Michael Mutter[1]

[1] FZI Research Center for Information Technology
Karlsruhe, Germany
{bock,mutter}@fzi.de
[2] Karlsruhe Institue of Technology
Karlsruhe, Germany
rudolph@kit.edu

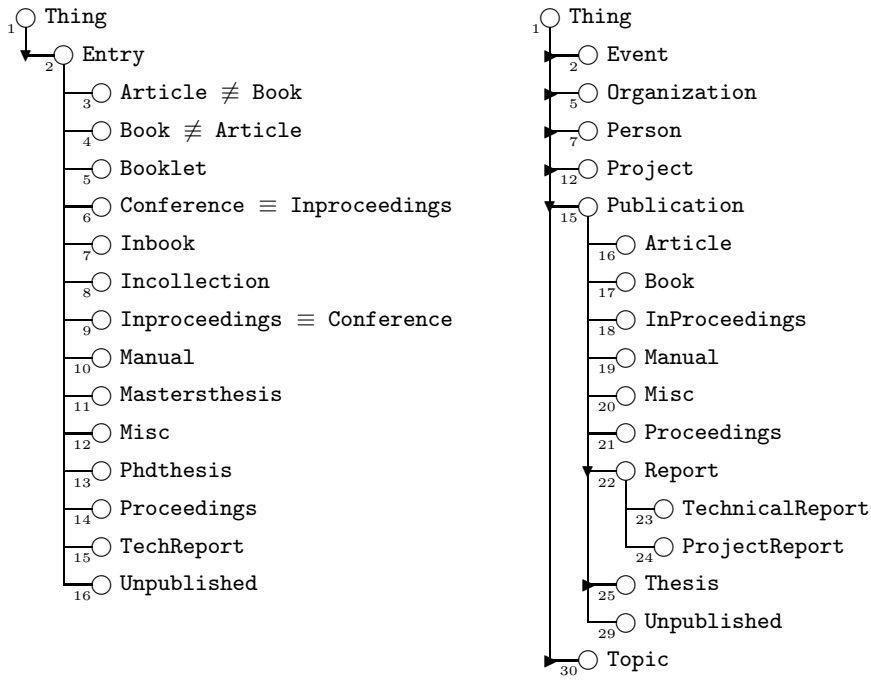
Abstract. Ontology alignment is a key challenge to allow for interoperability between heterogeneous semantic data sources. Today, most algorithms extract an alignment from a matrix of the pairwise similarities of ontological entities of two ontologies. However, this standard approach has severe disadvantages regarding scalability and is incapable of accounting for global alignment quality criteria that go beyond the aggregation of independent pairwise correspondence evaluations. This paper considers the ontology alignment problem as an optimisation problem that can be addressed using nature-inspired population-based optimisation heuristics. This allows for the deployment of an objective function which can be freely defined to take into account individual correspondence evaluations as well as global alignment constraints. Moreover, such algorithms can easily be parallelised and show anytime behaviour due to their iterative nature. The paper generalises an existing approach to the alignment problem based on discrete particle swarm optimisation, and presents a novel implementation based on evolutionary programming. First experimental results demonstrate feasibility and scalability of the presented approaches.

1 Introduction

Ontology alignment is a key challenge to allow for interoperability between heterogeneous semantic data sources. In particular for more expressive ontologies, the optimal alignment is difficult to discover, since correspondences cannot be assessed in isolation as they can interfere with each other on the alignment level [18]. For instance, two classes that are in a subclass relationship in an ontology cannot correspond to two disjoint classes in another ontology as this causes an incoherency in case a correspondence is interpreted as semantic equality. Figure 1 shows two ontologies that might be candidates for being aligned. In this example, an alignment containing the correspondences a:`Article` ↔ b:`Article` and a:`Book` ↔ b:`Publication` would cause such an incoherency.

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 71–90, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il



(a) MIT BibTEX ontology (modified excerpt) (b) Karlsruhe BibTEX ontology (modified excerpt)

Fig. 1. Example ontologies from the bibliography domain. Only class hierarchies and additional axioms for equivalence (≡) and disjointness (≢) are illustrated. Numbers at class nodes represent entity indices.

While most alignment systems disregard such global constraints and compute all pairwise entity similarities to extract an alignment from that matrix [22,12], there have been recent approaches that consider global quality criteria during the alignment computation process [21,4]. These systems, however, have problems with scalability when the ontologies become large, due to their exhaustive search algorithm for solving the constraint satisfaction problem.

We propose an approach that considers the ontology alignment problem an optimisation problem, and applies nature-inspired, population-based optimisation techniques for solving it. The ontology alignment problem is considered in a holistic manner, *i.e.* by taking local (correspondence level) and global (alignment level) quality criteria into account. Apart from this holistic consideration of ontology alignment, such optimisers are straightforwardly parallelisable [24,3] and show anytime behaviour, which makes them highly feasible for large scale alignment problems.

In previous work the ontology alignment problem was tackled by a *Particle Swarm Optimisation* algorithm [2]. This paper abstracts from this particular

oarieli@mta.ac.il

metaheuristic and demonstrates the general applicability of population-based optimisation techniques. Following this generalisation, an alternative metaheuristic, *Evolutionary Programming*, is introduced for solving the ontology alignment problem. Two prototype alignment systems are presented and evaluated for the two approaches, resp.

In the following Sect. 2 notations of ontology alignment and population-based optimisation are introduced, including an analysis of the search space for the ontology alignment problem. Section 3 discusses ontology alignment as a holistic optimisation problem and how it can be encoded in order to be tackled by optimisation algorithms. Two population-based optimisation algorithms are applied for the ontology alignment problem in Sect. 4, and evaluated in Sect. 5. Section 6 discusses related work before the paper concludes in Sect. 7.

2 Preliminaries

This work applies methods from the field of computational intelligence to a problem in the field of knowledge representation. This section introduces the basic notions of the problem domain *ontology alignment*, and of the *population-based optimisation* techniques used for solving it.

2.1 Ontology Alignment

The problem of ontology alignment is a rather generic mapping task and not necessarily restricted to a specific ontology language like OWL. Abstracting as much as possible from specificities of representation formalisms, we formalise the notions of ontology and ontology alignment.

Definition 1. *An* ontology $\mathcal{O}$ *is a set of statements*[1] *that are referring to a set of* ontology entities $\{e_1, \ldots, e_n\}$, *also called the* vocabulary *of the ontology and denoted by* $\mathsf{voc}(\mathcal{O})$. *Moreover, each* $e_i \in \mathsf{voc}(\mathcal{O})$ *is associated an* entity type $\tau(e_i)$ *with* $\tau : \mathsf{voc}(\mathcal{O}) \to \mathcal{T}$ *where* $\mathcal{T}$ *is a fixed finite set of types. The* size *of an ontology is the number of entities it contains:* $\sharp\mathcal{O} = |\mathsf{voc}(\mathcal{O})|$. *Moreover, for any* $t \in \mathcal{T}$ *we let* $\sharp_t\mathcal{O} = |\{e \mid e \in \mathsf{voc}(\mathcal{O}), \tau(e) = t\}|$ *denote the number of ontology entities of a specific type* t.

This definition accounts for the fact that many ontology formalisms support multi-sorted vocabularies. For instance, in the case of OWL, we have $\mathcal{T} = \{\texttt{class}, \texttt{object_property}, \texttt{data_property}, \texttt{individual}\}$. The subdivision of the ontology into these different categories is important since we assume that two entities of two ontologies can only be interlinked by a correspondence if they have the same type, as made more precise in the subsequent definition.

[1] Note that we do not further specify, how exactly the statements (also referred to as axioms) of an ontology look like.

oarieli@mta.ac.il

Definition 2. *Given two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, a* correspondence *between $\mathcal{O}_1$ and $\mathcal{O}_2$ is a pair of entities $C = \langle e, f \rangle$, where $e \in \mathsf{voc}(\mathcal{O}_1)$ and $f \in \mathsf{voc}(\mathcal{O}_2)$ and $\tau(e) = \tau(f)$, i.e., e and f are of the same entity type.*

Given two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, an alignment *$A \subseteq \mathsf{voc}(\mathcal{O}_1) \times \mathsf{voc}(\mathcal{O}_2)$ between $\mathcal{O}_1$ and $\mathcal{O}_2$ is a set of correspondences between $\mathcal{O}_1$ and $\mathcal{O}_2$, such that*

- *for each $e \in \mathsf{voc}(\mathcal{O}_1)$ there is at most one $f \in \mathsf{voc}(\mathcal{O}_2)$ with $\langle e, f \rangle \in A$ and*
- *for each $f \in \mathsf{voc}(\mathcal{O}_2)$ there is at most one $e \in \mathsf{voc}(\mathcal{O}_1)$ with $\langle e, f \rangle \in A$.*

The size *of an alignment, $|A|$, is the number of correspondences it contains.*

Note that the condition in definition 2 restricts the size of an alignment to be less or equal the size of the smaller ontology.

To determine the size of the problem space, *i.e.* the number of possible alignments between two ontologies, we first consider the number of possible alignments $\mathcal{A}$ between two sets M and N of entities all having the same type. As already stated, such an alignment can contain between 0 and $\min(|M|, |N|)$ correspondences. Let us consider the number of alignments with exactly k correspondences. To fix one such alignment, we have to pick k elements from N and – independently – k elements from M. For each of these choices there are $k!$ different ways of arranging the picked sets into k non-overlapping pairs from $M \times N$. This leaves us with

$$\binom{|M|}{k} \cdot \binom{|N|}{k} \cdot k! = \frac{|M|!}{(|M| - k)!} \binom{|N|}{k} \tag{1}$$

alignments of size k. To obtain the count of all alignments of arbitrary size we have to sum over the alignment numbers for all possible k.

Finally, to determine the total number of alignments between two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, we have to calculate the number of possible alignments for each type and obtain the number of possible combinations by multiplying the respective single-type alignment numbers. Thus we obtain for the number of total alignments:

$$\prod_{t \in \mathcal{T}} \sum_{k=0}^{\min(\sharp_t \mathcal{O}_1, \sharp_t \mathcal{O}_2)} \frac{\sharp_t \mathcal{O}_1!}{(\sharp_t \mathcal{O}_1 - k)!} \binom{\sharp_t \mathcal{O}_2}{k} \tag{2}$$

Thereby we obtain that the size of the problem space grows exponentially with the size of the ontologies.[2] This clearly implies that exhaustively searching the problem space for the optimal solution is infeasible for real-life ontologies.

2.2 Population-Based Optimisation Heuristics

The discipline of *Computational Intelligence* comprises various research categories out of which *Evolutionary Computation* [24] and *Computational Swarm*

[2] To be precise, we have to exclude the pathological case in which for every $t \in \mathcal{T}$ one of $\sharp_t \mathcal{O}_1$ or $\sharp_t \mathcal{O}_2$ remains constant.

oarieli@mta.ac.il

Intelligence [14] are two relatively young areas that have already gained notable impact in several application domains.

This class of algorithms can be regarded in terms of their characteristics of being population-based metaheuristics, which have shown to be efficient for complex optimisation problems. In that respect they share the following characteristics.

Definition 3. *Let $\mathbb{P}$ be the problem space associated to some problem. Then, a* population *is defined as a pair $\langle I, p\rangle$ where I is a finite set of* individuals *and $p : I \to \mathbb{P}$ is a function assigning to every individual of the population a position in the problem space.*

An update *operation is a function U mapping populations to populations.*

An optimisation run *with respect to U is a finite sequence $\langle I_1, p_1\rangle, \ldots, \langle I_n, p_n\rangle$ of populations, where $\langle I_{i+1}, p_{i+1}\rangle = U(\langle I_i, p_i\rangle)$ for all $i \in \{1, \ldots, n-1\}$. The numbers $1, \ldots, n$ are normally referred to as* generations.

The generic execution of a population-based optimisation algorithm starts with an initialisation step, where the population $\langle I_1, p_1\rangle$ is initialised. Following this, the algorithm iterates through its generations, where in each generation i the individuals are evaluated according to the objective function. For each individual this evaluation results in a fitness score in the problem space. Furthermore, in each iteration, the update operator U creates a new population for the next generation, based on the current population's fitness scores. This process is repeated until a termination criterion is fulfilled, *e.g.* a maximum number of generations n or a prescribed fitness score is reached.

The algorithm-specific update function is one of the most distinguishing features of each paradigm. *E.g.* algorithms might require the size of I_i to remain constant throughout generations, *i.e.* for all $i, j \in \{1, \ldots, n\}$ holds $|I_i| = |I_j|$.

The remainder of this section presents the two population-based optimisation heuristics *Particle Swarm Optimisation* and *Evolutionary Programming*.

Particle Swarm Optimisation. Commonly classified under the category of computational swarm intelligence, *Particle Swarm Optimisation* [15,23] has a strong emphasis on the social aspects of individuals (particles) in the population (swarm). After initialisation, the population remains constant, *i.e.* no particle leaves or joins the swarm and thus $\forall i \in \{1, \ldots, n-1\} : I_{i+1} = I_i$. In its classical implementation, every particle remembers the best position in the problem space it has ever visited (*personal best*), and the swarm remembers the best position any particle has ever visited (*global best*). In each iteration i (generation[3]) the update operation applies a velocity vector $v_i(x_j)$ to the position $p_i(x_j)$ of each particle $x_j \in I_i, 1 \leq j \leq |I_i|$, changing its position in the problem space. The velocity vector composes of the personal and global best positions, as well as an inertia component [23]. The new population in the $(i+1)$th iteration thus

[3] Since the swarm consists of the same set of particles throughout the execution of the algorithm, the term "generation" is slightly misplaced in this context.

oarieli@mta.ac.il

is $\langle I_{i+1}, p_{i+1}\rangle = U(\langle I_i, p_i\rangle) = \langle I_i, q_i\rangle$, where $\forall x_j \in I_i, 1 \leq j \leq |I_i| : q_i(x_j) = p_i(x_j) + v_i(x_j)$.

Evolutionary Programming. As one paradigm classified under the category of Evolutionary Algorithms, *Evolutionary Programming* [10,24] simulates species (individuals) that compete with each other in the problem space. As opposed to Genetic Algorithms, the update function in Evolutionary Programming does not involve a recombination operation for exchanging information between individuals in the population. However, each species creates an offspring by mutation, temporarily doubling the size of the population.

Formally, the update operation U is defined as follows: In a temporary population $\langle I_{i'}, p_{i'}\rangle$, for each $j \in \{1, \ldots, |I_i|\}$ let $x_{|I_i|+j} \in I_{i'}$ be the mutated species created by x_j. Thus, for all $k \in \{1, \ldots, |I_{i'}|\}$

$$p_{i'}(x_k) = \begin{cases} p_i(x_k) & \text{if } 1 \leq k \leq |I_i|, \\ q_i(x_k) & \text{if } (|I_i| + 1) \leq k \leq |I_{i'}| \end{cases}$$

where q_i maps a species to the new position after mutation.

In a subsequent selection step, half of that population becomes extinct, returning to the original size of the population. Survivors are typically determined by some sort of tournament selection, where species pairwise compete with other species, which results in a ranking that is used to select survivors of that generation. Formally, using a particular selection principle, the population in the $(i+1)$th generation is $\langle I_{i+1}, p_{i+1}\rangle$, such that $I_{i+1} \subseteq I_{i'}$ with $|I_{i+1}| = |I_i|$.

3 Ontology Alignment: A Holistic Optimisation Problem

Typical state-of-the-art ontology alignment systems follow the approach of pairwise entity comparison in order to generate alignments, as shown in Fig. 2. After some preprocessing operations, various basic matchers are applied to all $\sharp_t\mathcal{O}_1 \cdot \sharp_t\mathcal{O}_2$ pairs of entities from $\mathcal{O}_1$ and $\mathcal{O}_2$, for every $t \in \mathcal{T}$. The measures delivered by the basic matchers are combined and from the resulting similarity matrix an alignment is extracted by solving the resulting assignment problem. Typically this is done by variations of the Hungarian method [16]. In that approach correspondences are considered independently and the validity of the alignment with respect to any global constraints, such as inter-correspondence dependencies, can only be assessed after an actual alignment is extracted. Some systems deal with this situation by revisiting the similarity matrix when encountering the violation of such global constraints. The *ASMOV* system, for instance, applies a "semantic verification" by iteratively correcting the alignment [12].

The computation of the similarity matrix, alignment extraction, and iterative correction can become very costly when the ontologies are large. This section presents a holistic view on the alignment problem in terms of assessing the quality of an alignment in a way it cannot be assessed by considering its correspondences in isolation.

oarieli@mta.ac.il

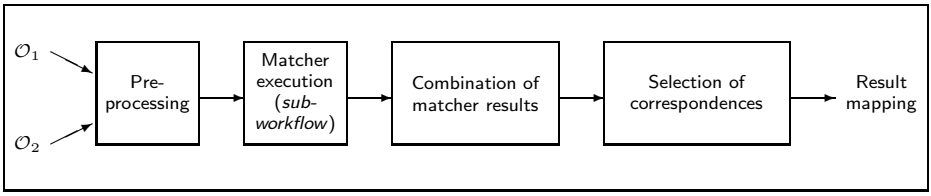


Fig. 2. General workflow for pairwise ontology matching (following Rahm [22])

3.1 Holistic Objective Function

A population-based optimisation algorithm as defined in Sect. 2.2 strives for finding the optimal argument for an objective function F, such that it becomes minimal (or maximal). In the case of ontology alignment, assume the objective function is minimal, iff an alignment is optimal. Thus the goal is to find

$$A^* = \text{argmin}_{A \in \mathbb{P}} \; F(A) \tag{3}$$

for all possible candidate alignments A from the space $\mathbb{P}$ of possible alignments.

The objective function F evaluates an alignment as a whole, thus it goes beyond the pairwise evaluation of entity pairs, *i.e.* candidate correspondences. We can thus subdivide F into a *local* and a *global* part: $F(A) = F_{\text{loc}}(A) + F_{\text{glob}}(A)$ which are described in more detail in the following.

Local Aspects. Some criteria of whether correspondences should be contained in an alignment that can be evaluated for each correspondence in isolation, *i.e.* there is a function $f : \mathsf{voc}(\mathcal{O}_1) \times \mathsf{voc}(\mathcal{O}_2) \to \mathbb{R}$ assigning to every correspondence $\langle e_1, e_2 \rangle$ a local score $f(e_1, e_2)$. The total contribution F_{loc} of the local aspects to the objective function is then obtained by summing over the local scores of all entity pairs of the given alignment:

$$F_{\text{loc}}(A) = \sum_{\langle e_1, e_2 \rangle \in A} f(e_1, e_2)$$

Those local aspects comprise lexical and linguistic similarity measures of entity labels, comments, and other annotations that might be available depending on the vocabularies and ontology languages used. This is what most approaches aggregate into similarities collected in a similarity matrix of entity pairs, and what Niepert *et al.* [21] refer to as a-priori likelihoods.

Global Aspects. Some alignment quality criteria go beyond the superposition of independent evaluations on the correspondence level. We mention the most natural possibilities and for some examples show how we incorporate them into our objective function, for a comprehensive description of the objective function see our previous work [2].

oarieli@mta.ac.il

Alignment Size. Typically there is not a complete overlap of ontologies. Thus it is important to determine the optimal size of an alignment. Since we naturally prefer larger alignments to smaller ones, we add a term $-\eta|A|$ to the objective function where η is a positive weighting factor.

Correspondence Correlations. Considering alignments from a structural point of view, correspondences can be evaluated differently depending on the presence of other correspondences in the alignment.

- Subsumption hierarchy: A correspondence between two classes is more likely to be correct, if the alignment also contains a correspondence of the classes' respective superclasses. The same holds for properties. If we denote the intra-ontological subclass/subproperty relationship with $\sqsubseteq$, the corresponding term of F_{glob} can be expressed by

$$\sum_{\langle e_1, e_2\rangle \in A} \sum_{\substack{\langle e_1', e_2'\rangle \in A \\ e_1 \sqsubseteq e_1', e_2 \sqsubseteq e_2'}} f(e_1', e_2').$$

  Note that although this term is represented as a sum over all correspondences it cannot be described locally, since the single summands again depend on the whole alignment A.
- Domain / range: A correspondence between two classes is more likely to be correct, if the alignment also contains correspondences between the properties of which these classes appear in the domain or range restriction. Likewise, a correspondence between two properties is more likely to be correct, if the alignment also contains correspondences between the classes that appear in their domain (or range) restrictions.
- Assertion axioms: A correspondence between two classes is more likely to be correct, if the alignment also contains correspondences between the individuals that are assigned to these classes. The same holds for property correspondences and correspondences of individuals asserted to the respective properties.

Alignment Coherence. Assuming a strict semantics of an alignment, it can cause incoherencies with respect to the ontologies it aligns [18]. The alignment criterion of coherency can only be assessed when considering the alignment as a whole.

3.2 Solution Representation

In the case of ontology alignment, each individual in the population represents a solution to the alignment problem, *i.e.* a candidate alignment. In order to efficiently converge towards the global optimum, an alignment must be represented such that it allows for

1. efficient manipulations of an alignment via the update operation U
2. efficient computation of a fitness score w.r.t. the objective function F.

oarieli@mta.ac.il

In the following, two representations are presented, the *correspondence set* representation as the straightforward solution, and the *correspondence permutation* representation as an alternative, which tends to be more efficient.

Correspondence Set. The *correspondence set* representation is the straightforward solution that regards an alignment naturally in the way it is defined in definition 2, namely as a set of correspondences. This representation allows for easy assessment of the alignment regarding the objective function. However, it requires the update operation U to be aware of global alignment constraints, such that only valid candidate alignments are produced by U. In this case, for instance, only 1:1 alignments are considered, and thus U must be specified in a way that only valid 1:1 alignments are generated when updating an individual.

Correspondence Permutation. The *correspondence permutation* representation denotes a novel data structure that overcomes the problem of the correspondence set representation.

Definition 4. *For two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, let m be the number of entities of type t in $\mathcal{O}_1$, and let n be the number of entities of type t in $\mathcal{O}_2$. A* correspondence permutation *is a function $\pi : \{1, \ldots, m\} \to \{1, \ldots, n\} \cup \{\square\}$, such that for all $1 \leq i < j \leq m$ with $\pi(i), \pi(j) \neq \square$ holds $\pi(i) \neq \pi(j)$.*

Intuitively, a correspondence permutation can be understood as an array of m elements: $\pi(1)\ \pi(2)\ \cdots\ \pi(m)$. Thereby each of the numbers $1 \ldots n$ occurs at most once in this array, whereas $\square$ can occur arbitrarily often.

Assuming that the array index represents the entity index of (the smaller) ontology $\mathcal{O}_1$ and that the array element at position i denotes the index $\pi(i)$ of the entity in ontology $\mathcal{O}_2$, it is straightforward that each array entry represents a correspondence $\langle e_i, f_{\pi(i)} \rangle$ if $f_{\pi(i)} \neq \square$, and no correspondence for e_i otherwise. Figure 3 shows an example of a correspondence permutation that represents a possible alignment for the two ontologies from Fig. 1. The array index represents the entity index of the ontology from Fig. 1a, while the array elements represent entity indices of the ontology from Fig. 1b.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	15	16	17	$\square$	$\square$	$\square$	$\square$	18	19	$\square$	20	25	21	23	29

Fig. 3. Example of a correspondence permutation using indices from Fig. 1

Since by definition 2 correspondences can only exist between entities of the same type, a solution representation would require $|\mathcal{T}|$ correspondence permutations for the different entity types. Following definition 3, let $\langle I, p \rangle$ be a population. Then for each $x \in I$, $p(x) = \pi_x$.

If for any $t \in \mathcal{T}$, $\sharp_t \mathcal{O}_1 \neq \sharp_t \mathcal{O}_2$, the ontologies should be swapped such that $m < n$ for increasing memory efficiency.

oarieli@mta.ac.il

4 The Charm of Nature-Inspired Optimisation Heuristics

The two examples of population-based optimisation techniques presented in Sect. 2.2 have been applied to the ontology alignment problem and implemented in two prototypes[4]:

- Ontology **Map**ping using **P**article **S**warm **O**ptimisation [2] (*MapPSO*)
- Ontology **Map**ping using **Evo**lutionary Programming (*MapEVO*).

Both algorithms share the same objective function which is composed of various base matchers that evaluate correspondences with respect to local and global characteristics. While the former cover lexical and linguistic similarities of entity labels and comments, the latter comprises structural similarities that evaluate correspondences in the context of the whole alignment. More precisely this means that the quality of a correspondence is assessed according to the presence of other correspondences in the alignment, as discussed in Sect. 3.1.

4.1 Ontology Mapping Using Particle Swarm Optimisation

A discrete particle swarm optimisation algorithm has been developed in order to tackle the ontology alignment problem [2]. The algorithm maintains a population of particles, each representing a valid alignment using the *correspondence set* representation. Since the problem space $\mathbb{P}$ is discrete, the classical notion of velocity fails, since it is tailored for continuous optimisation problems[5]. This approach picks up the notion of velocity as proposed by Correa *et al.* [5] and refines it in several respects in order to facilitate a more direct convergence towards a stable optimal alignment [2].

4.2 Ontology Mapping Using Evolutionary Programming

An evolutionary programming algorithm has been developed in order to tackle the ontology alignment problem. The algorithm utilises a population of species, each representing a valid alignment using the *correspondence permutation* representation. The approach does not incorporate a social component in terms of a velocity influenced by other individuals in the population. Instead it eliminates bad solutions and reproduces good ones in periodic selection processes. In order to avoid that only few species dominate the population, this selection process is not done in every generation. Experiments have shown that for the maximum number of generations n and population size $|I|$, the best results are obtained when conducting the selection process every $n/|I|$ generations, then eliminating a fraction of $1/|I|$ of the species and allowing the same number to reproduce themselves. Let s be a selection function transforming a set of species

[4] Both prototypes are available at `http://sourceforge.net/projects/mappso/`.

[5] In traditional (continuous) particle swarm optimisation, a velocity vector is added to the current position of a particle in each iteration in order to change its position in the search space.

oarieli@mta.ac.il

into another one, periodically as described above. There are two update operations u_s and u_e, such that $U(\langle I_i, p_i\rangle) = \langle I_{i+1}, p_{i+1}\rangle$ where $I_{i+1} = s(I_i)$ and $p_{i+1} = (u_e \circ u_s)(p_i)$. u_s and u_e are defined for the correspondence permutation data structure as follows. For the sake of simplicity let us consider only entities of type t. Let $n = \sharp_t \mathcal{O}_1$ and $m = \sharp_t \mathcal{O}_2$. Without loss of generality let $m < n$.

- The *switch* operator u_s transforms a correspondence permutation π into a new one π' by picking two indices i and j, such that $1 \leq i < j \leq m$ and $\pi(i), \pi(j) \neq \square$. Subsequently it transforms the correspondence permutation as follows by setting $\pi'(i) = \pi(j)$ and $\pi'(j) = \pi(i)$ as well as $\pi'(k) = \pi(k)$ for all $k \notin \{i, j\}$.

  Each index is selected for taking part in a switch with probability $P_\mathrm{s} = 1 - x$, where $x \in [0, 1]$ is the evaluation of correspondence $\langle e_i, f_{\pi(i)}\rangle$ in its global alignment context. Given that an evaluation of 1 is best and an evaluation of 0 is worst, this decreases the chance for good correspondences to be switched.
- The *exchange* operator u_e also transforms a correspondence permutation π into a new one π'. Let $R = \{1, \ldots, n\} \setminus \{\pi(1), \ldots, \pi(m)\}$ an archive of entity indices of $\mathcal{O}_2$ not currently participating in a correspondence. For each $i \in \{1, \ldots, m\}$, $\pi' = u_e(\pi)$ such that

$$\pi'(i) = \begin{cases} \square & \text{with probability } P_\square \text{ if } \pi(i) \neq \square, \\ k \in R & \text{with probability } P_\mathrm{fill} \text{ if } \pi(i) = \square, \\ k \in R & \text{with probability } P_\mathrm{ex} \text{ if } \pi(i) \neq \square, \\ \pi(i) & \text{otherwise} \end{cases}$$

  The exchange probabilities $P_\square$, P_fill, and P_ex depend on the evaluation of correspondence $\langle e_i, f_{\pi(i)}\rangle$ in its global alignment context, as well as the progress of the optimisation run [20]. The latter criterion, for instance, decreases P_fill in order to reduce the chance of new correspondences being created towards the end of the optimisation run.

Figure 4 illustrates the application of the *switch* operator, followed by an application of the *exchange* operator. The example uses entity indices from the ontologies in Fig. 1.

4.3 Advantages of Population-Based Optimisation Heuristics

Apart from the holistic alignment evaluation, the use of population-based optimisation heuristics has several other advantages. Firstly, they are inherently parallelisable [24], since individuals can be evaluated independently on parallel computing nodes. The *MapPSO* prototype has been successfully deployed to the Amazon Web Services™ (AWS)[6] cloud computing service [3].

Anytime behaviour is another feature of population-based optimisation heuristics. An optimisation run can be interrupted at any time returning the best solution found so far. In runtime-critical alignment scenarios, this might be a desirable property.

[6] http://aws.amazon.com/

oarieli@mta.ac.il

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	15	16	21	□	□	□	□	18	24	□	20	□	17	23	29

⇓

1	15	16	**17**	□	□	□	□	18	24	□	20	□	**21**	23	29

⇓

1	15	16	17	□	□	□	□	18	**19**	□	20	**25**	21	23	29

Fig. 4. Example of the application of the *switch* operator followed by an application of the *exchange* operator

5 Evaluation

Evaluating ontology alignment systems is a non-trivial task, since it requires the existence of high quality benchmarks [1], *i.e.* reference alignments for given pairs of ontologies. Such gold standards, however, do not exist for large ontologies, since they require human effort in order to verify their correctness. The Ontology Alignment Evaluation Initiative (OAEI)[7], however, does provide benchmarks for small ontologies.

The evaluation presented in this section is thus split into two parts. On the one hand there is an evaluation of the alignment quality using the gold standards provided by OAEI test cases. This illustrates the ability of the proposed approach to produce alignments of good accuracy. On the other hand a scalability evaluation is provided to show the applicability of the approach to large scale ontologies, where no gold standard is available.

5.1 Alignment Quality

In order to measure the quality of ontology alignments, the metrics *precision*, *recall*, and *F-measure* as known from information retrieval are applied. To this end, given an alignment A and a reference alignment R, the *precision* of A w.r.t. R is defined as

$$\mathrm{prec}(A,\, R) = \frac{|A \cap R|}{|A|} \tag{4}$$

denoting the fraction of found correspondences that are correct. The *recall* of A w.r.t. R is defined as

$$\mathrm{rec}(A,\, R) = \frac{|A \cap R|}{|R|} \tag{5}$$

denoting the fraction of expected correspondences that were found. The *F-measure* metric is the harmonic mean of precision and recall.

The OAEI provides several tracks where reference alignments are available that have been verified by human experts. The largest of those manually created reference alignments is in the *anatomy* track with 1522 correspondences.

[7] http://oaei.ontologymatching.org/

oarieli@mta.ac.il

Table 1. Precision, recall, and F-measure scores for *MapPSO* in the OAEI benchmarks track (2008–2010), grouped for tests 1xx, 2xx, and 3xx, as well as the harmonic mean for all 111 tests [8,9]

	2008			2009			2010		
	prec	rec	f-meas	prec	rec	f-meas	prec	rec	f-meas
1xx	0.92	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
2xx	0.48	0.53	0.50	0.75	0.73	0.74	0.67	0.59	0.63
3xx	0.49	0.25	0.33	0.54	0.29	0.38	0.72	0.39	0.51
h-mean	0.51	0.54	0.52	0.64	0.59	0.61	0.68	0.60	0.64

OAEI Benchmarks. The OAEI *benchmarks* track consists of 111 pairs of ontologies. One of the ontologies to be aligned is the same in all tests cases, while the other one is systematically altered by reducing features that are available for alignment systems [9, Sect. 3.1].

This section presents results for the prototypes *MapPSO* and *MapEVO* in the OAEI benchmarks track. The *MapPSO* system has participated in the track since 2008 and constantly increased its performance. Table 1 shows the precision, recall, and F-measure scores for each year.

Due to its novelty the *MapEVO* system has not officially participated in any previous OAEI campaign yet, however it has been tested using the latest 2010 benchmarks data set. The results of *MapEVO* compared to *MapPSO* in the 2010 OAEI benchmarks are shown in Table 2. The results were obtained using 16 species and 400 iterations for *MapEVO* and 60 particles and 250 iterations for *MapPSO*. These numbers were empirically determined such that further increasing them would not result in better scores.

As Table 2 illustrates, applying two different population-based optimisation heuristics results in roughly the same evaluation scores, with *MapEVO* performing slightly better, particularly for the 2xx tests. For both *MapPSO* and *MapEVO* the same objective function, *i.e.* base matcher configuration and aggregation was used, and both approaches score similar at the same tests. This shows that the alignment quality is largely influenced by the design of the objective function, while two different optimisation heuristics find the same optimum (according to this objective function) for each test.

Table 2. Precision, recall, and F-measure scores for *MapEVO*, *MapPSO* in the OAEI benchmarks track 2010, grouped for tests 1xx, 2xx, and 3xx, as well as the harmonic mean for all 111 tests. Results for *MapPSO* are taken from the official OAEI 2010 results.

	MapEVO			*MapPSO*		
	prec	rec	f-meas	prec	rec	f-meas
1xx	0.96	1.00	0.98	1.00	1.00	1.00
2xx	0.89	0.51	0.58	0.67	0.59	0.63
3xx	0.77	0.44	0.56	0.72	0.39	0.51
h-mean	0.89	0.53	0.66	0.68	0.60	0.64

oarieli@mta.ac.il

It is worth mentioning that in the OAEI 2009, symmetric precision and recall scores [7] were computed in addition to the classical scores. The track organisers reported very good results of *MapPSO* in terms of these relaxed metrics, which are comparable to the best participating systems [8, Sect. 3.2]. The reason for this might be the fact that metaheuristics are typically used to find a solution, which is very close to the optimum, followed by a local search step that fine-tunes this solution. *MapPSO* does currently not apply such a local search step, which results in close-to-optimal results, which are honoured by the relaxed evaluation metrics, but penalised using the classical ones.

OAEI Directory. The OAEI *directory* track comprises a collection of 4639 single test cases that represent segments of large web directories [9, Sect. 6.1].

The *MapPSO* system participated in this track in 2010 and scored similar to the best performing system. In the history of this track since 2006, *MapPSO* achieved the fourth highest scores among all participating systems [9, Sect. 6.2]. The reference alignment for this track is not publicly available, so a comparative evaluation with the *MapEVO* prototype could not be conducted.

OAEI Anatomy. The OAEI *anatomy* track provides two ontologies from the biomedical domain. The reference alignment for these real-world ontologies has been manually created by domain experts and contains 61 % of trivial correspondences that do not require the use of a domain-specific thesaurus [9, Sect. 4.1].

The anatomy ontologies have been processed by the *MapEVO* prototype outside of the OAEI context. Using the same objective function as in the OAEI benchmarks with 4 species and 5000 iterations, *MapEVO* scored with a precision of 0.82, recall of 0.42, and F-measure of 0.56. The high precision score demonstrates the ability to approach a solution that is of high quality. Since the objective function used in this experiment did not make any use of an external biomedical thesaurus[8] only a portion (43 %) of the correspondences could be identified, which is close to the 61 % of trivial correspondences that can be found without external knowledge.

5.2 Scalability

In order to demonstrate the ability to process large ontologies, an alignment of the Gene Ontology (GO)[9] with ∼31,650 classes, and the Medical Subject Headings (MeSH)[10] with ∼15,340 classes was computed using the *MapPSO* prototype. The experiment has been executed using the Amazon Web Services™ (AWS) cloud infrastructure [3]. Since no reference alignment is available for these data sets, no thorough evaluation of the alignment quality could be provided. However the convergence of the algorithm could be measured in terms of the fitness

[8] Many participants utilise the UMLS thesaurus (`http://www.nlm.nih.gov/research/umls/`) for this purpose.

[9] `http://www.geneontology.org/`

[10] `http://www.nlm.nih.gov/mesh/meshhome.html`

oarieli@mta.ac.il

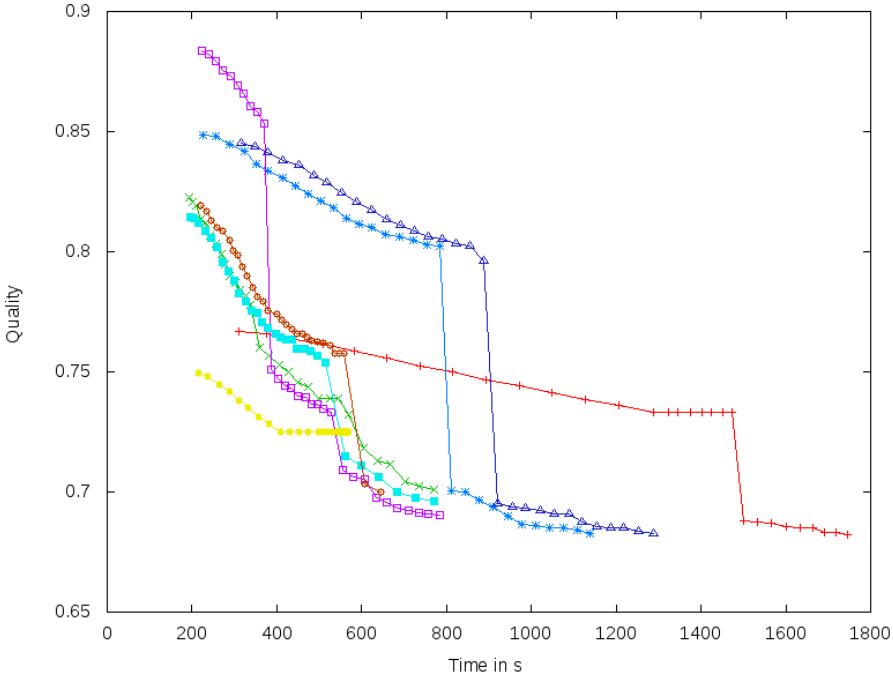


Fig. 5. Convergence behaviour regarding of the *MapPSO* algorithm (30 iterations) regarding alignment quality when aligning GO with MeSH. Each curve represents the quality of the alignment of a particular particle. Quality is denoted as a distance measure, thus lower values denote better quality.

scores of each particle throughout the iterations. Figure 5 shows the convergence behaviour of the *MapPSO* algorithm for this experiment. It can be observed that the quality of each alignment represented by a particle improves throughout the iterations. The case where a particle makes an adjustment towards the global best can be seen as a big step in a single iteration. In Fig. 6 the convergence towards the optimal alignment size is illustrated. Since an asynchronous update method was used [3] particles are progressing at a different pace, which is mostly depending on the alignment size of a particular particle.

5.3 Discussion

The fact that there are no reliable alignment gold standards available for large ontologies gave reason to split the evaluation in two: an *effectiveness* evaluation, and a *scalability* evaluation. The effectiveness evaluation takes well-established

oarieli@mta.ac.il

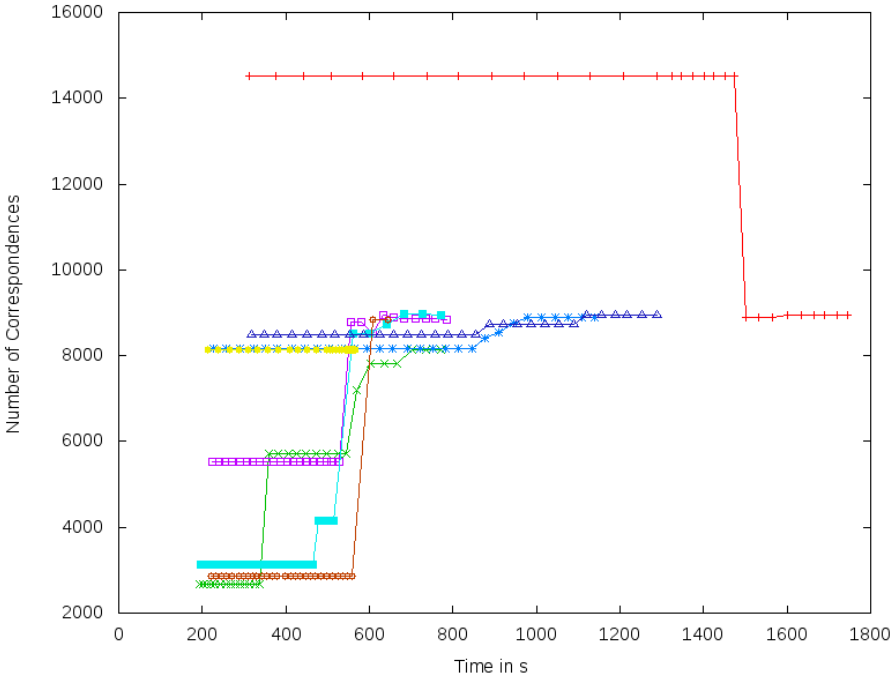


Fig. 6. Convergence behaviour of the *MapPSO* algorithm (30 iterations) regarding alignment size when aligning GO with MeSH. Each curve represents the size of an alignment of a particular particle.

OAEI data sets and demonstrates that the proposed systems *MapPSO* and *MapEVO* are capable of producing meaningful results. Comparing the systems with other OAEI participants shows, that other systems perform better in several test cases, particularly in the *benchmarks* data set[11]. The reason for this is, that the OAEI tests are small enough that an exhaustive, pairwise comparison of all entity pairs is feasible. In these cases there is no need to apply heuristic approaches, which are (without a local search component) inherently approximate. Experiences from OAEI participation in the years 2008–2010 indicate that the quality of results strongly depends on the specification of the objective function. Extending and tuning the objective function was the main reason for the increase in alignment quality from 2008 to 2010.

On the other hand, the scalability experiment demonstrates the potential of using population-based metaheuristics since they are capable of exploiting parallel computing infrastructures naturally. Consequently they are able to compute results for two large ontologies.

[11] Notably, the *CODI* system that implements a similar holistic alignment optimisation approach shows similar results in the *benchmarks* data set.

oarieli@mta.ac.il

6 Related Work

Few systems have previously been developed that allow for optimisation based on a holistic assessment of an alignment. Niepert *et al.* [21] apply Markov logic in order to express hard and soft constraints that characterise desired alignments. Their approach is implemented in the *CODI* system. Given the ontologies and a set of these constraints, the most probable alignment is computed using *maximum a-posteriori inference* by *integer linear programming*. The authors claim that their approach is the first to be able to avoid incoherence in alignments during the actual alignment process. This is an advantage over other systems where incoherence is reduced after an alignment is generated, which usually requires several iterations in order to find the best alignment without incoherencies, as *e.g.* in the *ASMOV* system [12]. A disadvantage of the *CODI* system is that it requires the computation of a large matrix of basic similarities in order to provide valuable a-priori likelihoods for correspondences to be correct. Moreover, the calculation of the most probable alignment is expensive and thus does not scale well for large ontologies.

The same scalability problem has been observed when utilising the constraint-based Answer Set Programming paradigm for ontology merging [4]. In that approach the linguistic features of lightweight taxonomies are exploited using a WordNet oracle in order to bring two taxonomies in a new merged structure. The Answer Set Programming solver processes a programme that declaratively describes constraints an alignment has to fulfil, and generates several answer sets with each one representing a solution according to the constraints.

Global alignment evaluation has been partially included to other state-of-the-art alignment systems, such as *RiMOM* [17], in terms of variations of the similarity flooding algorithm by Melnik *et al.* [19]. A recent global alignment evaluation metric regarding the alignment's structural preservation properties has been introduced by Joslyn *et al.* [13], and implemented for example in the *AgreementMaker* system [6]. In spite of using the structural global alignment metric, the system does only apply it in a selection step after computing several similarity matrices using local correspondence evaluation metrics.

Coming from the other direction, there are examples of successful applications of population-based optimisation heuristics to problems that are similar to the ontology alignment problem. Correa *et al.* [5], for instance, apply discrete particle swarm optimisation to the problem of feature selection for a Bayesian classifier. The quality of the classifier can only be assessed when considering all selected features at once, making the isolated evaluation of single features infeasible.

7 Conclusion

This paper presented a novel approach to tackle the problem of ontology alignment by utilising population-based optimisation heuristics. This overcomes the limitation of traditional alignment systems, which compute alignments by considering similarity measures purely on the correspondence level. We consider

oarieli@mta.ac.il

ontology alignment as a holistic optimisation problem that takes into account those local aspects (on the correspondence level), as well as global aspects (on the alignment level). The latter comprise inter-correspondence relationships, alignment size, coherency constraints, as well as other domain-specific global constraints that can be encoded in the objective function. Related approaches [21,4] have a scalability issue when solving the global constraint satisfaction problem.

We present two prototypes based on particle swarm optimisation (*MapPSO*) and evolutionary programming (*MapEVO*), resp., in order to demonstrate the feasibility of population-based optimisation heuristics in ontology alignment. Evaluation results show that the prototypes are able to find good quality alignments on benchmark data sets. The two optimisation approaches *MapPSO* and *MapEVO* compute similar alignments (regarding quality) using the same objective function. This shows that approaching the gold standard strongly depends on the objective function that is used, and thus leaves room for improvement regarding its details w.r.t. the benchmark data set. More generally this means that a wisely chosen objective function (w.r.t. a particular alignment scenario) can achieve good results using the proposed algorithms. In a different experiment, a straight convergence of the algorithm can be observed when aligning large ontologies. Despite the lack of a reference alignment for this scenario, the evaluation results allow the conclusion that with a wisely chosen objective function, population-based optimisation can be used to align large ontologies.

Future work on this topic will be directed towards the utilisation of ant colony optimisation algorithms, as well as the exploration of possibilities to adjust the objective function in order to maximise precision or recall depending on the alignment scenario. Additionally, means for user interrogation during the alignment process could be integrated to dynamically fine-tune the objective function. In the case of particle swarm optimisation, it has been shown [11] that users can also directly be involved in "guiding" the swarm to promising areas in the search space. Another extension of the proposed algorithms is to carry out a final local search step after the metaheuristic has found a suitable near-optimal solution. This will fine-tune the results while still avoiding exhaustive searches.

Acknowledgement. The presented research was partially funded by the German Federal Ministry of Economics (BMWi) under the project "Theseus" (number 01MQ07019).

References

1. Bellahsene, Z., Bonifati, A., Duchateau, F., Velegrakis, Y.: On Evaluating Schema Matching and Mapping. In: Bellahsene, Z., Bonifati, A., Rahm, E. (eds.) Schema Matching and Mapping. Springer, Heidelberg (2011)
2. Bock, J., Hettenhausen, J.: Discrete Particle Swarm Optimisation for Ontology Alignment. Information Sciences (2010)
3. Bock, J., Lenk, A., Dänschel, C.: Ontology Alignment in the Cloud. In: Proc. of the 5th Int. Workshop on Ontology Matching. CEUR Workshop Proceedings, vol. 689, pp. 73–84 (2010), http://ceur-ws.org

oarieli@mta.ac.il

4. Bock, J., Topor, R., Volz, R.: Ontology Merging using Answer Set Programming and Linguistic Knowledge. In: Proc. of the 2nd Int. Workshop on Ontology Matching. CEUR Workshop Proceedings, vol. 304, pp. 301–305 (2007), http://ceur-ws.org
5. Correa, E.S., Freitas, A.A., Johnson, C.G.: A New Discrete Particle Swarm Algorithm Applied to Attribute Selection in a Bioinformatics Data Set. In: Proc. of the 8th Genetic and Evolutionary Computation Conf. ACM, New York (2006)
6. Cruz, I.F., Antonelli, F.P., Stroe, C.: Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods. In: Shvaiko, P., Euzenat, J., Giunchiglia, F., Stuckenschmidt, H., Noy, N., Rosenthal, A. (eds.) Proceedings of the 4th International Workshop on Ontology Matching. CEUR Workshop Proceedings, vol. 551, pp. 49–60 (2009), http://ceur-ws.org
7. Ehrig, M., Euzenat, J.: Relaxed Precision and Recall for Ontology Matching. In: Proc. of the K-CAP Workshop on Integrating Ontologies. CEUR Workshop Proceedings, vol. 156, pp. 25–32 (2005), http://ceur-ws.org
8. Euzenat, J., Ferrara, A., Hollink, L., Isaac, A., Joslyn, C., Malaisé, V., Meilicke, C., Nikolov, A., Pane, J., Sabou, M., Scharffe, F., Shvaiko, P., Spiliopoulos, V., Stuckenschmidt, H., Šváb-Zamazal, O., Svátek, V., Trojahn, C., Vouros, G., Wang, S.: Results of the Ontology Alignment Evaluation Initiative. In: Proc. of the 4th Int. Workshop on Ontology Matching. CEUR Workshop Proceedings, vol. 551, pp. 73–126 (2009), http://ceur-ws.org
9. Euzenat, J., Ferrara, A., Meilicke, C., Nikolov, A., Pane, J., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Šváb-Zamazal, O., Svátek, V., Trojahn dos Santos, C.: Results of the Ontology Alignment Evaluation Initiative. In: Proc. of the 5th Int. Workshop on Ontology Matching. CEUR Workshop Proceedings, vol. 689, pp. 85–117 (2010), http://ceur-ws.org
10. Fogel, M.J., Owens, L.J., Walsh, A.J.: Artificial Intelligence through Simulated Evolution. Wiley, Chichester (1966)
11. Hettenhausen, J.: Interactive Multi-Objective Particle Swarm Optimisation with Heatmap Visualisation based User Interface. Master's thesis, Griffith University (2007)
12. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology Matching with Semantic Verification. Web Semantics 7(3), 235–251 (2009)
13. Joslyn, C.A., Paulson, P., White, A.: Measuring the Structural Preservation of Semantic Hierarchy Alignments. In: Shvaiko, P., Euzenat, J., Giunchiglia, F., Stuckenschmidt, H., Noy, N., Rosenthal, A. (eds.) Proceedings of the 4th International Workshop on Ontology Matching. CEUR Workshop Proceedings, vol. 551, pp. 61–72 (2009), http://ceur-ws.org
14. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann (April 2001)
15. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proc. of IEEE Int. Conf. on Neural Networks, vol. 4. IEEE Computer Society (1995)
16. Kuhn, H.W.: The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly 2(1-2), 83–97 (1955)
17. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. IEEE Transactions on Knowledge and Data Engineering 21(8), 1218–1232 (2009)
18. Meilicke, C., Stuckenschmidt, H.: Repairing Ontology Mappings. In: Proc. of the 22nd AAAI Conf. on Artificial Intelligence. AAAI Press (2007)

oarieli@mta.ac.il

19. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Georgakopoulos, D., Agrawal, R., Dittrich, K. (eds.) Proceedings of the 18th International Conference on Data Engineering, pp. 117–128. IEEE Computer Society, Washington, DC, USA (2002)
20. Mutter, M.: Ontology Alignment durch Evolutionäre Algorithmen. Diplomarbeit, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany (2011)
21. Niepert, M., Meilicke, C., Stuckenschmidt, H.: A Probabilistic-Logical Framework for Ontology Matching. In: Proc. of the 24th AAAI Conf. on Artificial Intelligence. AAAI Press (2010)
22. Rahm, E.: Towards Large-Scale Schema and Ontology Matching. In: Bellahsene, Z., Bonifati, A., Rahm, E. (eds.) Schema Matching and Mapping. Springer, Heidelberg (2011)
23. Shi, Y., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: Proc. of IEEE Int. Conf. on Evolutionary Computation. IEEE Computer Society (1998)
24. Whitley, D.: An overview of evolutionary algorithms: practical issues and common pitfalls. Information and Software Technology 43(14), 817–831 (2001)

oarieli@mta.ac.il

Handling Preferences in P2P Systems

Luciano Caroprese and Ester Zumpano

DEIS, Univ. della Calabria, 87030 Rende, Italy
{lcaroprese,zumpano}@deis.unical.it

Abstract. A peer to peer system easily provides a way to aggregate information distributed in the network. Anyhow, while collecting data it is quite natural for a source peer to associate different degrees of reliability to the portion of data provided by its neighbor peers. This paper investigates the data exchange problem among distributed independent sources and concentrates on the task of using dynamic preferences to drive the integration process in the case of conflicting information. Previous works in the literature are *rigid* in the sense that preferences between conflicting sets of atoms, that a peer can import, only depends on the priorities associated to the source peers at design time. These approaches do not allow to model concepts such as *"import tuples from the peer having the highest upload speed if they conflict"* or *"among conflicting values import the most recent ones"*. In this paper it is supposed the existence of a special peer, called *authority peer*. It contains information about the peers in the network, is accessible from each peer of the system and is used to enhance preference mechanism. The framework, here proposed, ensures dynamism by allowing to select among different scenarios looking at the properties of data provided by the peers: this is done by *"dynamically"* establishing priorities among mapping rules.

1 Introduction

Modeling cooperation between independent sources is a very old problem that emerges in all the areas in which data must be exchanged between a source and a target. This paper provides a contribution to the effort of ranking and returning the preferred information in *Peer-to-Peer* (P2P) data sharing systems. Each peer, joining a P2P system, can both provide or consume data and has information about its neighbors, that is about the peers that are reachable and can provide data of interest. In a P2P system each peer exhibits a set of *mapping rules*, that is a set of semantic correspondences to its neighbor peers. A mapping rule transfers data from a source peer to a target peer. The entry of a new peer in the system is extremely simple as it just requires the definition of its mapping rules. By using mapping rules a peer can participate and access data available in its neighborhood, and through its neighborhood it becomes accessible to all the other peers in the system. Therefore, due to the specified mechanism for collecting data, a query, that can be posed to any peer in the system, is answered by using locally stored data and all the information that can be imported, without corrupting the local database, from its neighbors.

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 91–106, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

This paper uses the following interpretation for mapping rules: it is allowed to import from other peers only atoms not violating integrity constraints. Under this semantics preferred scenarios are those in which peers import maximal sets of facts not violating integrity constraints.

The framework, here proposed, considers a central peer, also called *authority peer*, that acts as an authority. Specifically, it monitors the system and acquires from each peer joining the network information such as download speed, timestamp, etc.

The information collected by the authority peer into a table, called *authority table*, is accessible from each peer of the system and is used to enhance preference mechanism in selecting information when conflicting data arise.

The presence of a 'central' peer is problematic in a real implementation of a P2P system, which is a 'decentralized' environment. Therefore, even if for the sake of simplicity of presentation we assume the existence of just one authority peer, the framework, here presented, can be immediately extended allowing each peer to be an authority peer. In other words, each peer can be though (becomes) an authority peer as soon as it exposes an authority table.

More specifically, this paper associates different levels of priority to mapping rules, that is different degree of reliability to data a peer can import. This ensures dynamism as it allows to select among different scenarios looking at the properties of data provided by the peers.

Previous works in the literature are *rigid* in the sense that preferences between conflicting sets of atoms, that a peer can import, only depends on the priorities associated to the source peers at design time. These approaches [3,4,12,14,16] do not allow to model concepts such as *"import tuples from the peer having the highest upload speed if they conflict"* or *"among conflicting values import the most recent ones"*.

This task is achieved as following: each peer, when queried, constructs a preference list of its mapping rules. The order in which mapping rules are activated, determines, in the case of conflicting information the selection of one scenario w.r.t. another one. Note that, even though each peer knows its own mapping rules, the order in which they will work (and therefore the data that will be preferred in the case of conflicting information) depends on the information provided by the authority peer at run time.

Organization. The rest of the paper is structured as follows. Section 2 introduces backgrounds. Section 3 is devoted to describing a new semantics for P2P systems, named weak model semantics. Section 4 describes two alternative characterizations of previously presented semantics. The first one captures the preferred weak model semantics, using a logic program with priorities ([11,12]); the second one, here presented, captures the trusted weak model semantics by means of a logic program with weak constraints. The interesting result is that both characterizations use the same logic program that basically generates all weak models. Finally, Section 5 reports preliminary results on computational complexity, concluding remarks and directions for further research.

oarieli@mta.ac.il

2 Background

Familiarity is assumed with deductive database [1], logic programming and computational complexity [17,20]. Now we briefly recall some notation used in the rest of the paper. A *normal rule* or ($\vee - free$ - disjunction free) is of the form $H \leftarrow \beta$, where H is an atom (*head* of the rule) and β is a conjunction of literals (*body* of the rule). A *normal program* is a finite set of normal rules. Given a program $\mathcal{P}$, $ground(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} ground(r)$, where $ground(r)$ denotes the set of all rules obtained from the rule r by replacing variables with constants in all possible ways. An *interpretation* is a set of ground atoms. The *semantics* of a program $\mathcal{P}$ is given by the set of its *stable models*, denoted as $\mathcal{SM}(\mathcal{P})$. A normal positive program $\mathcal{P}$ has one stable model, that coincides with the *least model*. It will be also denoted as $\mathcal{LM}(\mathcal{P})$. Given a database $\mathcal{DB}$ and a relation symbol r, $\mathcal{DB}[r]$ denotes the set of atoms whose relation symbol is r.

2.1 Prioritized Logic Programs

Various forms of priorities have been investigated in logic languages [6,22]. We refer to the extension proposed in [22]. Given two atoms e_1 and e_2, the statement $e_2 \succeq e_1$ is a *priority* stating that for each a_2 instance of e_2 and for each a_1 instance of e_1, a_2 has higher priority than a_1. If $e_2 \succeq e_1$ and $e_1 \not\succeq e_2$ we write $e_2 \succ e_1$. If $e_2 \succ e_1$ the sets of instantiations of e_1 and e_2 must be disjoint. The relation $\succeq$ is transitive and reflexive. A *prioritized logic program* $\mathcal{P}_\Phi$ is a pair $(\mathcal{P}, \Phi)$ where $\mathcal{P}$ is a program and Φ is a set of priorities. Φ^* denotes the set of priorities which can be reflexively or transitively derived from Φ. Given a PLP $(\mathcal{P}, \Phi)$, the relation $\sqsupseteq$ is defined over the stable models of $\mathcal{P}$. For any stable models M_1, M_2 and M_3 of $\mathcal{P}$: (i) $M_1 \sqsupseteq M_1$; (ii)$M_2 \sqsupseteq M_1$ if a) $\exists\ e_2 \in M_2 - M_1$, $\exists\ e_1 \in M_1 - M_2$ such that $(e_2 \succeq e_1) \in \Phi^*$ and b) $\nexists\ e_3 \in M_1 - M_2$ such that $(e_3 \succ e_2) \in \Phi^*$; (iii) if $M_2 \sqsupseteq M_1$ and $M_1 \sqsupseteq M_3$, then $M_2 \sqsupseteq M_3$. If $M_2 \sqsupseteq M_1$ holds, then we say that M_2 is *preferable* to M_1 w.r.t. Φ. Moreover, we write $M_2 \sqsupset M_1$ if $M_2 \sqsupseteq M_1$ and $M_1 \not\sqsupseteq M_2$. An interpretation M is a *preferred* stable model of $(\mathcal{P}, \Phi)$ if M is a stable model of $\mathcal{P}$ and there is no stable model N s.t. $N \sqsupset M$. $\mathcal{PSM}(\mathcal{P}_\Phi)$ denotes the set of preferred stable models of $\mathcal{P}_\Phi$.

2.2 Integrity Constraints

A *(strong) integrity constraint ic* expresses a relationship among data that every database instance is required to satisfy, and it is expressed by a rule of the form $\leftarrow \beta$, where β is a conjunction of literals. A database $\mathcal{DB}$ satisfies *ic* if $\mathcal{DB} \not\models \beta$. A *weak integrity constraint* [8,19] is a variant of an integrity constraint and is expressed by a rule of the form: $\Leftarrow \beta\ [l]$, where l is a positive integer, called *level*, expressing the priority of the constraint. Weak constraints express desiderata, that is *conditions to be possibly satisfied* and their informal semantics is to minimize the number of violated instances. Given a program $\mathcal{P}$, a set of strong constraints $\mathcal{SC}$ and a set weak constraints $\mathcal{WC}$, an interpretation M is a stable model of $\mathcal{P} \cup \mathcal{SC} \cup \mathcal{WC}$ if M is a stable model of $\mathcal{P}$ that (i) satisfies

oarieli@mta.ac.il

$\mathcal{SC}$ and (ii) minimizes the number of unsatisfied instances of weak constraints in $\mathcal{WC}$ according to their importance, that is it minimizes the number of violated weak constraints in the highest priority level[1], then minimizes the number of the violated weak constraints in the next lower level, and so on and so forth [2]. Formally, we associate to each priority level $i \in [1..n]$ a positive weight $f(i)$ inductively defined as follows: $f(1) = 1$, $f(i) = f(i-1) * |ground(\mathcal{WC})| + 1$, $1 < i \leq n$, where $|ground(\mathcal{WC})|$ is the total number of ground instances of weak constraints in $\mathcal{WC}$. Further, given a stable model M for a program $\mathcal{P}$, the *objective function* $H_{\mathcal{P},\mathcal{M}}$ is defined as: $H_{\mathcal{P},\mathcal{M}} = f(1) * N_1^M + \ldots + f(n) * N_n^M$, where N_i^M is the number of ground instances of weak constraints whose priority level is i which are violated in M $(1 \leq i \leq n)$. Therefore, among the stable models of $\mathcal{P}$ that satisfies $\mathcal{SC}$ we select those minimizing the objective function. Observe that, the objective function guarantees that the violation of a single constraint of priority level i is more *expensive* than the violation of *all* weak constraints of the lower levels (i.e. all levels $< i$).

3 A New Semantics for P2P Systems

A *(peer) predicate symbol* is a pair $i : p$, where i is a *peer identifier* and p is a predicate symbol. A *(peer) atom* is of the form $i : A$, where i is a *peer identifier* and A is a standard atom. A *(peer) literal* is a peer atom $i : A$ or its negation $not\ i : A$. A conjunction $i : A_1, \ldots, i : A_m, not\ i : A_{m+1}, \ldots, not\ i : A_n, \phi$, where ϕ is a conjunction of built-in atoms[3], will be also denoted as $i : \beta$, with β equals to $A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n, \phi$.

A *(peer) rule* can be of one of the following three types:

1. STANDARD RULE. It is of the form $i : H \leftarrow i : \beta$, where $i : H$ is an atom and $i : \beta$ is a conjunction of atoms and built-in atoms.
2. INTEGRITY CONSTRAINT. It is of the form $\leftarrow i : \beta$, where $i : \beta$ is a conjunction of literals and built-in atoms.
3. MAPPING RULE. It is of the form $i : H \hookleftarrow j : \beta$, where $i : H$ is an atom, $j : \beta$ is a conjunction of atoms and built-in atoms and $i \neq j$.

In these mapping rules $i : H$ is called *head*, while $i : \beta$ (resp. $j : \beta$) is called *body*. Negation is allowed just in the body of integrity constraints. The operator $ground(\cdot)$ is defined is standard way. Whenever the peer is clear from the context, its identifier can be omitted. The definition of a predicate $i : p$ consists of the set of rules whose heads contain predicate symbol $i : p$. A predicate can be of three different kinds: *base predicate*, *derived predicate* and *mapping predicate*. A base predicate is defined by a set of ground facts; a derived predicate is defined by a set of standard rules and a mapping predicate is defined by a set of mapping rules.

[1] Higher values for priority levels mark weak constraints of higher importance.

[2] A program with a weak constraint $\Leftarrow p(X)$ can be regarded as modeling a minimization problem whose objective function is the cardinality of p.

[3] A *built-in atom* is of the form $\theta(X, Y)$, where X and Y are terms and θ is a comparison predicate.

oarieli@mta.ac.il

An atom $i : p(X)$ is a *base atom* (resp. *derived atom*, *mapping atom*) if $i : p$ is a base predicate (resp. standard predicate, mapping predicate). Given an interpretation M, $\mathcal{DB}(M)$ (resp. $\mathcal{LP}(M)$, $\mathcal{MP}(M)$) denotes the subset of base atoms (resp. derived atoms, mapping atoms) in M. Given a mapping rule $r = i : H \hookleftarrow j : \beta$, $St(r)$ denotes the corresponding standard logic rule $i : H \leftarrow j : \beta$. Analogously, given a set of mapping rules $\mathcal{MP}$, $St(\mathcal{MP}) = \{St(r) \mid r \in \mathcal{MP}\}$.

We first recall the concept of satisfaction of mapping rules in [11,12]. Given an interpretation M, an atom H and a conjunction of atoms β: (i) $val_M(i : H \leftarrow i : \beta) = val_M(i : H) \geq val_M(i : \beta)$, (ii) $val_M(i : H \hookleftarrow j : \beta) = val_M(i : H) \leq val_M(j : \beta)$ [4]. Thus, if the body is *true*, the head of a standard rule *must* be *true*, whereas the head of a mapping rule *could* be *true*.

The idea in [11,12] is that if the source peer satisfies the body of a mapping rule, the head can be inferred in the target peer only if it does not generate inconsistencies.

In [11,12] a P2P system is defined as a set of peers. A peer $\mathcal{P}_i$ is defined as a tuple $\langle \mathcal{DB}_i, \mathcal{LP}_i, \mathcal{MP}_i, \mathcal{IC}_i \rangle$ where $\mathcal{DB}_i$ is a set of atoms (the *local database*), $\mathcal{LP}_i$ is a set of standard rules, $\mathcal{MP}_i$ is a set of mapping rules and $\mathcal{IC}_i$ is a set of integrity constraints. $\mathcal{P}_i$ is assumed to be *locally consistent* that is $\mathcal{LP}_i \cup \mathcal{DB}_i \models \mathcal{IC}_i$. However, inconsistencies could be introduced when mapping atoms are imported into $\mathcal{P}_i$. Therefore, the mapping rules in $\mathcal{MP}_i$ are used to import into $\mathcal{P}_i$ maximal sets S of mapping atoms that do not generate violations of the integrity constraints in $\mathcal{IC}_i$, i.e. $\mathcal{LP}_i \cup \mathcal{DB}_i \cup S \models \mathcal{IC}_i$.

Previous frameworks [3,4,12,14,16] do not provide any mechanism to set priorities among mapping rules disallowing us to model concept as *"import tuples from the peer having the highest upload speed if they conflict"* or *"among conflicting values import the most recent ones"*. Here we extend the definition of *peer* and a *P2P system* with a framework that allows us to model those interesting scenarios.

Definition 1. P2P System. A *peer* $\mathcal{P}_i$ is a tuple $\langle \mathcal{DB}_i, \mathcal{LP}_i, \mathcal{MP}_i, \mathcal{PMP}_i, \mathcal{IC}_i \rangle$, where:

- $\mathcal{DB}_i$ is a set of atoms;
- $\mathcal{LP}_i$ is a set of standard rules. If $i > 0$ (i.e. the peer is not an authority peer) $\mathcal{LP}_i$ contains the set, called $\mathcal{LP}_i^{pref}$, of the following rules:
  - $i : pref(Relation, P_a, P_b) \leftarrow i : pref_m(Relation, P_a, P_b)$
  - $i : pref(Relation, P_a, P_b) \leftarrow i : pref_b(Relation, P_a, P_b)$
  - $i : pref(Relation, P_a, P_b) \leftarrow i : pref(Relation, P_a, P_c), i : pref(Relation, P_c, P_b)$

  where
  - $i : pref(R, P_a, P_b)$ expresses a preference. It states that it is *preferable* for $\mathcal{P}_i$, in case of conflicting data, to import a tuple into the relation R from the peer P_a than the peer P_b.
  - $i : pref_m(R, P_a, P_b)$ expresses a dynamic preference derived from the information imported from the authority peer.
  - $i : pref_b(R, P_a, P_b)$ expresses a static preference locally defined.

[4] $val_M(\cdot)$ returns the truth value of the argument w.r.t. M.

oarieli@mta.ac.il

- $\mathcal{MP}_i$ is a set of mapping rules of the form: $i : H \leftarrow j : \beta$, where $j \neq 0$;
- $\mathcal{PMP}_i$ is a set of mapping rules of the form:

$$i : pref_m(Relation, P_a, P_b) \leftarrow 0 : \beta$$

- $\mathcal{IC}_i$ is a set of integrity constraints over predicates defined by $\mathcal{DB}_i$, $\mathcal{LP}_i$ and $\mathcal{MP}_i$.

A *P2P system* $\mathcal{PS}$ is a set of peers $\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n\}$ where $\mathcal{P}_0$ is of the form $\langle \mathcal{DB}_0, \emptyset, \emptyset, \emptyset, \emptyset \rangle$ and $\mathcal{DB}_0$ only contains a relation, called *authority table*, reporting data about the peers whose schema is: $0 : info(Peer, Attribute_1, \ldots, Attribute_n)$. The peer $\mathcal{P}_0$ is called *authority peer*. □

Therefore, a peer $\mathcal{P}_i$ stores preferences among its neighbor peers in the relation $i : pref$. In more details, part of the preferences of $\mathcal{P}_i$ can be locally defined by means of tuples of $i : pref_b$ and part of them can be defined by means of tuples imported from the authority peer $\mathcal{P}_0$ that contains information about other peers. Note that, the derived predicate $i : pref$ is the *transitive closure* of the union of $i : pref_m$ and $i : pref_b$. Here we assume the presence of only one authority peer ($\mathcal{P}_0$) which has a very simple structure. Anyhow, each peer can act as an authority peer as soon as it exposes an *authority table*.

Example 1. Consider the P2P system $\mathcal{PS}$ containing the peers:

$$\mathcal{P}_0 = \langle \{info(2, 500), info(3, 200)\}, \emptyset, \emptyset, \emptyset, \emptyset \rangle,$$

$$\begin{aligned}\mathcal{P}_1 = \langle \emptyset, \mathcal{LP}_1^{pref}, \{&person(Id, Name) \leftarrow employee(Id, Name),\\ &person(Id, Name) \leftarrow people(Id, Name)\},\\ &\{pref_m(R, U, V) \leftarrow info(U, X), info(V, Y), X > Y\},\\ &\{\leftarrow person(Id, N_1), person(Id, N_2), N_1 \neq N_2\}\rangle,\end{aligned}$$

$$\mathcal{P}_2 = \langle \{employee(44768, tom), employee(98765, sam)\}, \mathcal{LP}_2^{pref}, \emptyset, \emptyset, \emptyset \rangle \text{ and}$$

$$\mathcal{P}_3 = \langle \{people(44768, dan), people(23765, lia)\}, \mathcal{LP}_3^{pref}, \emptyset, \emptyset, \emptyset \rangle.$$

The authority peer, $\mathcal{P}_0$, reports the values of peers' upload speed: for $\mathcal{P}_2$ it is 500 while for $\mathcal{P}_3$ it is 200.

The peer $\mathcal{P}_1$ contains two mapping rules allowing to import into the relation *person* tuples from the relation *employee* of the peer $\mathcal{P}_2$ and from the relation *people* of the peer $\mathcal{P}_3$. It defines a key constraint over the relation *person*: two different persons cannot have the same Id. Moreover, $\mathcal{P}_1$ defines a preference mapping rules stating that in case of conflicting data it is preferred to import tuples from the peer having the higher upload speed. □

The relation $i : pref(R, P_a, P_b)$ allows to partition the mapping rules belonging to $\mathcal{MP}_i$ into an ordered list $\langle \mathcal{MP}_{i,1}, \ldots, \mathcal{MP}_{i,k_i} \rangle$ of *priority levels* that establishes priorities among mapping rules. The intuitive meaning of this structure is

oarieli@mta.ac.il

that the data imported by means of a mapping rule r belonging to $\mathcal{MP}_{i,u}$ are 'more important' to data imported by means of a mapping rule s belonging to $\mathcal{MP}_{i,v}$ for each $u < v$.

Given a *P2P system* $\mathcal{PS} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n\}$ and an index $i \in [1..n]$ the relation $i : pref$, modeling the preferences of the peer $\mathcal{P}_i$, is obtained by the union of the relation $i : pref_m$, whose tuples are imported from the table $0 : info$ using the mapping rules $\mathcal{PMP}_i$, and the relation relation $i : pref_b$. Formally,
$i : pref = \mathcal{LM}(0 : info \ \cup \ i : pref_b \cup \ st(\mathcal{PMP}_i) \cup \mathcal{LP}_i^{pref})[i : pref]$
In other words, in order to populate the relation $i : pref$, the standard version of rules in $\mathcal{PMP}_i$ is evaluated over the relation $0 : info$. This process allows to infer the relation $i : pref_m$.

Observe that, evaluating the standard version of the mapping rules in $\mathcal{PMP}_i$ does not cause any violation of integrity constraints. Finally, the relation $i : pref$ is inferred by evaluating the rules in $\mathcal{LP}_i^{pref}$.

The relation $i : pref$ of a peer $\mathcal{P}_i$ is then used to classify its mapping rules.

Definition 2. *Let* $\mathcal{PS} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n\}$ *be a* P2P system*,* $i \in [1..n]$ *and* ρ_1, ρ_2 *mapping rules in* $\mathcal{MP}_i$ *of the form:*

- $\rho_1 = \ i : r \leftarrow v : \beta_1$ *and*
- $\rho_2 = \ i : r \leftarrow w : \beta_2$.

We say that ρ_1 is preferred over ρ_2 *and write* $\rho_1 > \rho_2$ *if the atom* $i : pref(r, v, w)$ *is* true *and the atom* $i : pref(r, w, v)$ *is* false*. Moreover, if* $\rho_1 \not> \rho_2$ *and* $\rho_2 \not> \rho_1$ *then* $\rho_1 \approx \rho_2$*. Given two sets of mapping rules* M *and* N*, we say that* M *is preferred over* N *and write* $M > N$ *if they are not empty and for each* $\rho_2 \in N$ *there exists* $\rho_1 \in M$ *and s.t.* $\rho_1 > \rho_2$*.*

A priority list $\mathcal{PL}_i$ *of a peer* $\mathcal{P}_i$ *is a list* $\langle \mathcal{MP}_{i,1}, \ldots, \mathcal{MP}_{i,k_i} \rangle$ *where:*

- *for each* $j \in [1..k_i - 1]$ *then* $\mathcal{MP}_{i,j} > \mathcal{MP}_{i,j+1}$*.*
- *for each* $j \in [1..k_i]$ *and* $\rho_1, \rho_2 \in \mathcal{MP}_{i,j}$ *then* $\rho_1 \approx \rho_2$*.*

We call the sets $\mathcal{MP}_{i,j}$ priority levels*.* □

Intuitively, a mapping rule can belong to a priority list of level i if it is less preferable of the mapping rules of the previous levels. Moreover, each level has to contain mapping rules that cannot be compared. These two conditions ensure that there is a unique way to build the priority lists.

Therefore, the following result holds.

Proposition 1. *Let* $\mathcal{PS} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n\}$ *be a* P2P system*. Each peer* $\mathcal{P}_i$*, with* $i \in [1..n]$*, admits exactly one* priority list $\mathcal{PL}_i$*.* □

Definition 3. *Given a peer* $\mathcal{P}_i = \langle \mathcal{DB}_i, \mathcal{LP}_i, \mathcal{MP}_i, \mathcal{PMP}_i, \mathcal{IC}_i \rangle$*, we define its* evaluated version *as* $\overline{\mathcal{P}_i} = \langle \mathcal{DB}_i, \mathcal{LP}_i, \langle \mathcal{MP}_{i,1}, \ldots, \mathcal{MP}_{i,k_i} \rangle, \mathcal{IC}_i \rangle$*. Given a* P2P system $\mathcal{PS} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n\}$*, we define its* evaluated version *as* $\overline{\mathcal{PS}} = \{\overline{\mathcal{P}_1}, \ldots, \overline{\mathcal{P}_n}\}$*.* □

oarieli@mta.ac.il

The first step to evaluate a query over a P2P system $\mathcal{PS}$ is to compute its evaluated version $\overline{\mathcal{PS}}$, that is to compute the priority list of mapping rules for each peer. Then, the inference process is performed over $\overline{\mathcal{PS}}$. The evaluated version of a P2P system is *dynamic* in the sense that the priority lists change each time the information provided by the authority peer changes, that is each time the authority table changes.

Example 2. Consider the P2P system in Example 1:
The evaluated version of $\mathcal{P}_1$ is:
$\overline{\mathcal{P}_1} = \langle \emptyset, \mathcal{LP}_1^{pref}, \langle \{person(Id,\ Name) \hookleftarrow employee(Id, Name)\}, \{person(Id, Name) \hookleftarrow people(Id, Name)\}\rangle, \{\leftarrow person(Id, N_1), person(Id, N_2), N_1 \neq N_2\}\rangle$.
The mapping rule that allows to import tuples from the peer $\mathcal{P}_2$ has an higher priority that the one that allows to import tuples from the peer $\mathcal{P}_3$. □

Note that, a change of the upload speed of a peer reflects in a change in the authority table. This variation will be easily acquired by each peer joining the system and reflects, as a consequence, in a change in its *priority list*: this let to a modification of the order in which mapping rules will be activated.

Definition 4. *Given a* P2P system $\mathcal{PS} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n\}$ *we define:*

- $\mathcal{DB}(\mathcal{PS}) = \bigcup_{i\in[1..n]} \mathcal{DB}_i$,
- $\mathcal{LP}\ (\mathcal{PS}) \ = \bigcup_{i\in[1..n]} \mathcal{LP}_i$,
- $\mathcal{MP}(\mathcal{PS}) = \bigcup_{i\in[1..n]} \mathcal{MP}_i$ *and*
- $\mathcal{IC}(\mathcal{PS}) = \bigcup_{i\in[1..n]} \mathcal{IC}_i$.

□

Previous definition allows to present the concept of *weak model*.

Definition 5. Weak Model [11,12]. Given a P2P system $\mathcal{PS}$, an interpretation M is a *weak model* for $\mathcal{PS}$ if M is the least model of $\mathcal{LP}(\mathcal{PS}) \cup \mathcal{DB}(\mathcal{PS}) \cup St(\mathcal{MP}(\mathcal{PS})^M) \cup \mathcal{IC}(\mathcal{PS})$, where $\mathcal{MP}(\mathcal{PS})^M$ is obtained from $ground(\mathcal{MP}(\mathcal{PS}))$ by removing all mapping rules whose head is *false* with respect to M.

The set of weak models for $\mathcal{PS}$ is denoted as $\mathcal{WM}(\mathcal{PS})$. □

Given a P2P system $\mathcal{PS}$, the set $\mathcal{WM}(\mathcal{PS})$ models all the ways the peers can import atoms from their neighbors without violating integrity constraints.

Example 3. Consider the P2P system $\mathcal{PS}$ containing the three peers:

$\mathcal{P}_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$,

$\mathcal{P}_1 = \langle \emptyset, \mathcal{LP}_1^{pref}, \{a \hookleftarrow c, b \hookleftarrow d\}, \emptyset, \{\leftarrow a, b\}\rangle$,

$\mathcal{P}_2 = \langle \{c\}, \mathcal{LP}_2^{pref}, \emptyset,\ \emptyset, \emptyset \rangle$ and

$\mathcal{P}_3 = \langle \{d\}, \mathcal{LP}_3^{pref}, \emptyset, \emptyset, \emptyset \rangle$.

oarieli@mta.ac.il

The authority peer does not store any data about the other peers in this case. The peer $\mathcal{P}_1$ contains two mapping rules but, as its set $\mathcal{PMP}_1$ and its relation $1 : pref_b$ are empty, it does not define any preference among them.
The weak models of $\mathcal{PS}$ are:

- $M_1 = \{a, c, d\}$
- $M_2 = \{b, c, d\}$
- $M_3 = \{c, d\}$.

Note that, M_1 and M_2 import respectively a and b, whereas no mapping atom is imported by M_3. □

We now recall the proposal of *Preferred Weak Models* given in [11,12] that following the reasonable principle of *"importing as much knowledge as possible"* selects among weak models those that contain maximal sets of mapping atoms.

Definition 6. Preferred Weak Model [11,12]. Let $\mathcal{PS}$ be a P2P system and M and N weak models of $\mathcal{PS}$. Then M is *preferable over* N with respect to $\mathcal{PS}$, denoted as $M \sqsupset N$, if $\mathcal{MP}(M) \supset \mathcal{MP}(N)$. A weak model M is said to be *preferred* if there is no weak model N such that $N \sqsupset M$. The set of *preferred weak models* for $\mathcal{PS}$ is denoted as $\mathcal{PWM}(\mathcal{PS})$. □

A preferred weak model is a weak model with a maximal set of mapping atoms.

Example 4. (Example 3 continued). The preferred weak models of $\mathcal{PS}$ are:

- $M_1 = \{a, c, d\}$
- $M_2 = \{b, c, d\}$

In fact, there are two different ways to import maximal sets of mapping atoms into $\mathcal{P}_1$: importing a using the mapping rule $a \hookleftarrow c$ or importing b using the mapping rule $b \hookleftarrow d$. □

Here we introduce a new semantics for P2P systems that enhances the one modeled by Definition 6 with priority levels among mapping rules in order to select the weak models containing a maximum number of mapping atoms according to their importance. We will call these weak models, *Trusted Weak Models*.

Definition 7. Mapping Atoms with Trusted Level. Let M be a weak model of a P2P system $\mathcal{PS} = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_n\}$ and $\overline{\mathcal{PS}} = \{\overline{\mathcal{P}_1}, \ldots, \overline{\mathcal{P}_n}\}$, where $\overline{\mathcal{P}_i} = \langle \mathcal{DB}_i, \mathcal{LP}_i, \langle \mathcal{MP}_{i,1}, \ldots, \mathcal{MP}_{i,k_i} \rangle, \mathcal{IC}_i \rangle$ with $i \in [1..n]$, its evaluated version.

We denote as $\mathcal{MP}(M, k)$ the set of mapping atoms a belonging to M such that k is the smallest index of a priority level that allows to infer a (by means of one of its mapping rules) that is:

$\mathcal{MP}(M, k) = \{i : a(t) \mid i : a(t) \in M \wedge\ k$ is the smallest index *s.t.* $i : a(t) \hookleftarrow \beta \in \mathcal{MP}_{i,k} \ \wedge\ M \models \beta$, for $i \in [1..n]\}$. □

Observe that, $\mathcal{MP}(M, k) \cap \mathcal{MP}(M, h) = \emptyset$ for each $k \neq h$.

oarieli@mta.ac.il

Definition 8. Trusted Weak Model. Let $\mathcal{PS}$ be a P2P system and M and N weak models of $\mathcal{PS}$. We say that M *is trustable over* N *with respect to* $\mathcal{PS}$, denoted as $M \succ N$, if there exists $j \geq 1$ such that for each $1 \leq i < j$ we have $|\mathcal{MP}(M,i)| = |\mathcal{MP}(N,i)|$ and $|\mathcal{MP}(M,j)| > |\mathcal{MP}(N,j)|$. We say that M is a *trusted weak model* of $\mathcal{PS}$ if there is no weak model N of $\mathcal{PS}$ such that $N \succ M$. The set of trusted weak models for $\mathcal{PS}$ is denoted as $\mathcal{TWM}(\mathcal{PS})$. □

Therefore, the new semantics first selects the weak models containing a maximum number of atoms inferred by means of mapping rule of level 1 (the highest level), then among that models those containing a maximum number of atoms inferred by means of mapping rule of level 2 and so on.

By definition, for each P2P systems $\mathcal{PS}$ the following relations hold:

- $\mathcal{PWM}(\mathcal{PS}) \subseteq \mathcal{WM}(\mathcal{PS})$
- $\mathcal{TWM}(\mathcal{PS}) \subseteq \mathcal{WM}(\mathcal{PS})$

Moreover, the interesting property that each trusted weak model is a preferred weak model holds.

Theorem 1. *Let $\mathcal{PS}$ be a P2P system. Then $\mathcal{TWM}(\mathcal{PS}) \subseteq \mathcal{PWM}(\mathcal{PS})$.* □

Summarizing, for each P2P $\mathcal{PS}$, the relations between weak, preferred weak and trusted weak models are:

$$\mathcal{TWM}(\mathcal{PS}) \subseteq \mathcal{PWM}(\mathcal{PS}) \subseteq \mathcal{WM}(\mathcal{PS})$$

Example 5. Consider the P2P system $\mathcal{PS}$ of Examples 1 and 2. We have two preferred weak models:

- $M_1 = \{person(\ 44768, tom), person(98765, sam), person(23765, lia)\}$
  $\cup \mathcal{DB}_2 \cup \mathcal{DB}_3$
- $M_2 = \{person(44768, dan), person(98765, sam), person(23765, lia)\}$
  $\cup \mathcal{DB}_2 \cup \mathcal{DB}_3$.

Observe that, $|\mathcal{MP}(M_1, 1)| = 1$, $|\mathcal{MP}(M_2,\ 1)| = 0$. Therefore, as the first mapping rule has priority w.r.t. to the second one (as it belongs to the level 1), M_1 is the unique trusted weak model of $\mathcal{PS}$. □

4 An Alternative Characterization of the New Semantics

This section describes two alternative characterizations of previously presented semantics. The first one has been proposed in [11,12] and captures the preferred weak model semantics by a logic program with priorities. Here, we present the second one that captures the trusted weak model semantics by means of a logic program with weak constraints. The interesting result is that both characterizations use the same logic program that basically generates all weak models. Preferred weak models are then obtained by adding to this program a set of

oarieli@mta.ac.il

priorities and trusted weak models are obtained by adding to it a set of weak constraints. We recall that, given a mapping rule $H \hookleftarrow \beta$, if β is *true* in the source peer then two mutually exclusive actions are possible in the target peer: *either importing* H *or not importing* H. This behavior can be modeled by a logic rule of the form: $H \oplus H' \leftarrow \beta$[5], where if $H = i : p(X)$ then $H' = i : p'(X)$. The meaning of this rule is that if β is *true* then exactly one atom between H and H' must be *true*. Therefore, while the head atom H models the *import* action, H' models the *not-import* action.

Definition 9. *Given a P2P system* $\mathcal{PS}$,

- $Rew(\mathcal{MP}(\mathcal{PS})) = \{H \oplus H' \leftarrow \beta | H \hookleftarrow \beta \in \mathcal{MP}(\mathcal{PS})\}$
- $Rew(\mathcal{PS}) = \mathcal{DB}(\mathcal{PS}) \cup \mathcal{LP}(\mathcal{PS}) \cup Rew(\mathcal{MP}(\mathcal{PS})) \cup \mathcal{IC}(\mathcal{PS})$

□

Given a stable model M for $Rew(\mathcal{PS})$, the subset of non-primed atoms of M is denoted by $\Omega(M)$. The operator $\Omega(\cdot)$ is extended to sets of models. The weak models of a P2P system $\mathcal{PS}$ can be obtained from the stable models of $Rew(\mathcal{PS})$.

Theorem 2. Given a P2P system $\mathcal{PS}$, $\mathcal{WM}(\mathcal{PS}) = \Omega(\ \mathcal{SM}(Rew(\mathcal{PS})))$. □

Prioritized programs and preferred stable models

The preferred weak model semantics can be computed by adding to the logic program in Definition 9 a set of priorities so that obtaining a prioritized logic program [6,22].

Definition 10. Given a P2P system $\mathcal{PS}$, (i) $\Phi(\mathcal{PS}) = \{H \succeq H' | H \hookleftarrow \beta \in \mathcal{MP}(\mathcal{PS})\}$, (ii) $Rew^P(\mathcal{PS}) = (Rew(\mathcal{PS}), \Phi(\mathcal{PS}))$. □

The priority statement $H \succeq H'$ in the previous definition intuitively stands that *it is preferable to import* H *w.r.t. not import it.*

The following theorem shows the equivalence between the preferred weak models of a P2P system $\mathcal{PS}$ and the preferred stable models of its rewriting $Rew^P(\mathcal{PS})$.

Theorem 3. Given a P2P system $\mathcal{PS}$, $\mathcal{PWM}(\mathcal{PS}) = \Omega(\mathcal{PSM}(Rew^P(\mathcal{PS})))$. □

Trusted weak models with stable model semantics

Trusted weak model semantics can be computed by adding to the logic program in Definition 9 a set of weak constraints [8,19].

Definition 11. Let $\mathcal{PS} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ be a P2P system and $\overline{\mathcal{PS}} = \{\overline{\mathcal{P}_1}, \ldots, \overline{\mathcal{P}_n}\}$ its *evaluated version*, where $\overline{\mathcal{P}_i} = \langle \mathcal{DB}_i, \mathcal{LP}_i, \langle \mathcal{MP}_{i,1}, \ldots, \mathcal{MP}_{i,k_i} \rangle, \mathcal{IC}_i \rangle$, for $i \in [1..n]$. Let l be the maximum priority level occurring in $\overline{\mathcal{PS}}$, that is $l = max(\{k_i | i \in [i..n]\})$, we define:

[5] $H \oplus H' \leftarrow \beta$ is just shorthand for $H \leftarrow \beta, not\ H'$ and $H' \leftarrow \beta, not\ H$.

oarieli@mta.ac.il

- $\mathcal{G}(\mathcal{PS}) = \{i : p^g(t) | i : p(t) \hookleftarrow \beta \in ground(\mathcal{MP}(\mathcal{PS})))\}$,
- $\mathcal{WC}(\mathcal{PS}) = \{\Leftarrow H^g, not\ H \quad [l-k+1] | H \hookleftarrow \beta \in ground(\mathcal{MP}_{i,k})$ for $i \in [1..n]\}$ where if $H = i : p(X)$ then $H^g = i : p^g(X)$,
- $Rew^T(\mathcal{PS}) = Rew(\mathcal{PS}) \cup \mathcal{G}(\mathcal{PS}) \cup \mathcal{WC}(\mathcal{PS})$. □

The weak constraint $\Leftarrow H^g, not\ H\ [l-k+1]$ in the previous definition intuitively expresses that the *desirable action of importing* H *is in the level* $l-k+1$. We recall that, as pointed out in the Section 2, the violation of a single weak constraint of priority level i is more expensive than the violation of all weak constraints of the lower levels. Therefore, our objective function asks for the minimization of violations starting from the highest priority level of weak constraints that correspond to the lowest level of mapping rules.

The following theorem shows the equivalence between the trusted weak models of a P2P system $\mathcal{PS}$ and the stable models of its rewriting $Rew^T(\mathcal{PS})$.

Theorem 4. Given a P2P system $\mathcal{PS}$, $\mathcal{TWM}(\mathcal{PS}) = \Omega(\mathcal{SM}(Rew^T(\mathcal{PS})))$. □

Example 6. Consider the P2P system $\mathcal{PS}$ of Example 1. $Rew(\mathcal{MP}(\mathcal{PS}))$ contains the following rules:

- $person(Id, Name) \oplus person'(Id, Name) \leftarrow employee(Id, Name)$
- $person(Id, Name) \oplus person'(Id, Name) \leftarrow people(Id, Name)$

The stable models of $Rew(\mathcal{PS}) = \mathcal{DB}(\mathcal{PS}) \cup \mathcal{LP}(\mathcal{PS}) \cup Rew(\mathcal{MP}(\mathcal{PS})) \cup \mathcal{IC}(\mathcal{PS})$ are:

- $M_1 = \{person(44768, tom), person'(44768, dan), person(98765, sam), person(23765, lia)\} \cup \mathcal{DB}_2 \cup \mathcal{DB}_3$
- $M_2 = \{person(44768, dan), person'(44768, tom), person(98765, sam), person(23765, lia)\} \cup \mathcal{DB}_2 \cup \mathcal{DB}_3$.
- $M_3 = \{person'(44768, dan), person'(44768, tom), person(98765, sam), person(23765, lia)\} \cup \mathcal{DB}_2 \cup \mathcal{DB}_3$.

The set of weak models obtained by removing the primed atoms, by using the operator $\Omega(\cdot)$, are:

- $W_1 = \{person(44768, tom), person(98765, sam), person(23765, lia)\} \cup \mathcal{DB}_2 \cup \mathcal{DB}_3$
- $W_2 = \{person(44768, dan), person(98765, sam), person(23765, lia)\} \cup \mathcal{DB}_2 \cup \mathcal{DB}_3$.
- $W_3 = \{person(98765, sam), person(23765, lia)\} \cup \mathcal{DB}_2 \cup \mathcal{DB}_3$.

$Rew^P(\mathcal{PS}) = (Rew(\mathcal{PS}), \Phi(\mathcal{PS}))$, where

$$\Phi(\mathcal{PS}) = \{person(Id, Name) \succeq person'(Id, Name)\}$$

oarieli@mta.ac.il

The preferred stable models of $Rew^P(\mathcal{PS})$ are M_1 and M_2. Therefore, the preferred weak models are W_1 and W_2.

$Rew^T(\mathcal{PS}) = Rew(\mathcal{PS}) \cup \mathcal{G}(\mathcal{PS}) \cup \mathcal{WC}(\mathcal{PS})$, where

- $\mathcal{G}(\mathcal{PS}) = \{person^g(44768, dan), person^g(44768, tom), person^g(98765, sam), person^g(23765, lia)\}$
- $\mathcal{WC}(\mathcal{PS}) = \{\Leftarrow person^g(\ 44768, tom), not\ person(44768, tom)[2],$
  $\Leftarrow person^g(98765, sam), not\ person(98765, sam)[2],$
  $\Leftarrow person^g(44768, dan), not\ person(44768, dan)[1],$
  $\Leftarrow person^g(23765, lia), notperson(23765, lia)[1]\}.$

The unique stable model of $Rew^T(\mathcal{PS})$ is M_1. Therefore, the unique trusted weak model is W_1. □

Observe that, the prioritized program $Rew^P(\mathcal{PS})$ cannot be processed directly using ASP solvers such as DLV [19] or Smodels [23]. Therefore, a more complex rewriting technique embedding priorities into a unique logic program has been presented in [13]. On the other hand, as the mechanism of weak constraints is implemented in ASP solvers, the program $Rew^T(\mathcal{PS})$ can be directly evaluated.

5 Discussion

5.1 Complexity Results

Complexity results for computing the *preferred weak models* can be immediately determined by considering analogous results on stable model semantics for prioritized logic program [6,22]. Complexity results for computing the *trusted weak models* can be immediately fixed by considering analogous results on stable model semantics for the fragment of DATALOG allowing negation ($\neg$), strong (s) and weak constraints with priorities ($w^{\succ}$), denoted as DATALOG$^{\neg,s,w^{\succ}}$ [8]. It is rather clear that strong constraints do not affect at all the computational complexity [6]. Additionally, in [8] it has been shown that adding weak constraints to a logic program do not cause an increase of the complexity as it always remains in the same level of polynomial hierarchy. Our framework models trusted weak models by using DATALOG$^{\neg,s,w^{\succ}}$, that is Datalog plus strong and weak constraints with priorities. In this case, deciding whether an atom is *true* in some trusted weak model increases from $\mathcal{NP}$ to Δ_2^P ([8]).

5.2 Related Works

The problem of integrating and querying databases in a P2P system has been investigated in [3,9,10,16,15]. In [9,10] a new semantics for a P2P system, based

[6] In fact, under stable model semantics a strong constraint of the form $\leftarrow \beta$ is actually a shorthand for $p \leftarrow \beta, \neg p$.

oarieli@mta.ac.il

on epistemic logic, is proposed. A peer collects data and constructs its epistemic theory. Epistemic logic ensures that each peer only exports the data it really knows, that is its *certain data*.

In [16,15] a characterization of P2P database systems and a model-theoretic semantics dealing with inconsistent peers is proposed. The basic idea is that if a peer does not have models all (ground) queries submitted to the peer are *true*. Thus, if some database is inconsistent it does not mean that the entire system is inconsistent.

The semantics in [16,15] coincides with the epistemic semantics in [9,10].

None of the previous proposals take into account the possibility of modeling some preference criteria while performing the data integration process.

A new interesting semantics for data exchange systems that goes in this direction has been recently proposed in [3,4]. This semantics allows for a cooperation among pairwise peers that related each other by means of data exchange constraints (i.e. mapping rules) and trust relationships. The decision by a peer on what other data to consider (besides its local data) does not depend only on its data exchange constraints, but also on the trust relationship that it has with other peers. For example, if peer $\mathcal{P}_1$ trusts peer $\mathcal{P}_2$'s data more than its own, then $\mathcal{P}_1$ will accommodate its data to $\mathcal{P}_2$ and will keep the data exchange constraints between them and its local integrity constraints satisfied while respecting its trust relationship (that is it will give priority to $\mathcal{P}_2$'s data with respect to its own data).

The main difference between our proposal and the recent proposal by Bertossi and Bravo is related to the different levels allowed for modeling priorities among peers. More specifically, in [4] trust relationships allows just modeling two different reliability levels w.r.t the database of a neighbor peer: $\mathcal{P}_1$ trusts itself *less* than $\mathcal{P}_2$ or $\mathcal{P}_1$ trusts itself the *same* as $\mathcal{P}_2$, whereas we can associate different priority levels to different portions of the database exported by the same peer. Moreover, our approach allows to import mapping atoms only if they do not produce a local violation of integrity constraints (a peer trusts its own data *more* than any other imported data), whereas in [4] a peer collects data from its neighbors using data exchange constraints and then applies a local repair semantics [2,18].

5.3 Concluding Remarks and Directions for Further Research

This paper investigates the data exchange problem among distributed independent sources and concentrates on the task of using dynamic preferences to drive the integration process in the case of conflicting information. Previous works in the literature are *rigid* in the sense that preferences between conflicting sets of atoms, that a peer can import, only depends on the priorities associated to the source peers at design time. These approaches do not allow to model concepts such as *"import tuples from the peer having the highest upload speed if they conflict"* or *"among conflicting values import the most recent ones"*. In this paper it is supposed the existence of a special peer, called *authority peer*. It contains information about the peers in the network, is accessible from each peer of the

oarieli@mta.ac.il

system and is used to enhance preference mechanism. The framework, here proposed, ensures dynamism by allowing to select among different scenarios looking at the properties of data provided by the peers: this is done by *"dynamically"* establishing priorities among mapping rules. Specifically, it identifies, among the set of all possible models, the trusted weak models, i.e. those importing the maximal sets of atoms according to specified preferences. The paper has provided preliminary results on computational complexity. Two important issues are left for further research. First, a deep investigation of the relationships of the present work with works on non-monotonic multi-context system, such as [5,7,21] and a more extended discussion on computational complexity.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: PODS, pp. 68–79 (1999)
3. Bertossi, L.E., Bravo, L.: Query Answering in Peer-to-Peer Data Exchange Systems. In: Lindner, W., Fischer, F., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268, pp. 476–485. Springer, Heidelberg (2004)
4. Bertossi, L.E., Bravo, L.: The Semantics of Consistency and Trust in Peer Data Exchange Systems. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 107–122. Springer, Heidelberg (2007)
5. Bikakis, A., Antoniou, G.: Distributed Defeasible Contextual Reasoning in Ambient Computing. In: Aarts, E., Crowley, J.L., de Ruyter, B., Gerhäuser, H., Pflaum, A., Schmidt, J., Wichert, R. (eds.) AmI 2008. LNCS, vol. 5355, pp. 308–325. Springer, Heidelberg (2008)
6. Brewka, G., Eiter, T.: Preferred answer sets for extended logic programs. Artificial Intelligence 109, 297–356 (1998)
7. Brewka, G., Eiter, T.: Equilibria in heterogeneous nonmonotonic multi-context systems. In: AAAI, pp. 385–390 (2007)
8. Buccafurri, F., Leone, N., Rullo, P.: Enhancing disjunctive datalog by constraints. IEEE Trans. Knowl. Data Eng. 12(5), 845–860 (2000)
9. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Inconsistency tolerance in p2p data integration: An epistemic logic approach. Inf. Syst. 33(4-5), 360–384 (2008)
10. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Logical foundations of peer-to-peer data integration. In: PODS, pp. 241–251 (2004)
11. Caroprese, L., Greco, S., Zumpano, E.: A logic programming approach to querying and integrating p2p deductive databases. In: FLAIRS Conference, pp. 31–36 (2006)
12. Caroprese, L., Molinaro, C., Zumpano, E.: Integrating and querying p2p deductive databases. In: IDEAS, pp. 285–290 (2006)
13. Caroprese, L., Zumpano, E.: Consistent Data Integration in P2P Deductive Databases. In: Prade, H., Subrahmanian, V.S. (eds.) SUM 2007. LNCS (LNAI), vol. 4772, pp. 230–243. Springer, Heidelberg (2007)
14. Caroprese, L., Zumpano, E.: Modeling Cooperation in P2P Data Management Systems. In: An, A., Matwin, S., Raś, Z.W., Ślęzak, D. (eds.) ISMIS 2008. LNCS (LNAI), vol. 4994, pp. 225–235. Springer, Heidelberg (2008)

oarieli@mta.ac.il

15. Franconi, E., Kuper, G.M., Lopatenko, A., Serafini, L.: A robust logical and computational characterisation of peer-to-peer database systems. In: DBISP2P, pp. 64–76 (2003)
16. Franconi, E., Kuper, G.M., Lopatenko, A., Zaihrayeu, I.: Queries and updates in the codb peer to peer database system. In: VLDB, pp. 1277–1280 (2004)
17. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP, pp. 1070–1080 (1988)
18. Greco, G., Greco, S., Zumpano, E.: A logical framework for querying and repairing inconsistent databases. IEEE Trans. Knowl. Data Eng. 15(6), 1389–1408 (2003)
19. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Trans. Comput. Log. 7(3), 499–562 (2006)
20. Papadimitriou, C.M.: Computational complexity. Addison-Wesley, Reading (1994)
21. Roelofsen, F., Serafini, L.: Minimal and absent information in contexts. In: Proc. 19th International Joint Conference on Artificial Intelligence, IJCAI 2005 (2005)
22. Sakama, C., Inoue, K.: Prioritized logic programming and its application to commonsense reasoning (2000)
23. Syrjänen, T., Niemelä, I.: The Smodels System. In: Eiter, T., Faber, W., Truszczyński, M. (eds.) LPNMR 2001. LNCS (LNAI), vol. 2173, pp. 434–438. Springer, Heidelberg (2001)

oarieli@mta.ac.il

Backing and Undercutting in Abstract Argumentation Frameworks

Andrea Cohen, Alejandro J. García, and Guillermo R. Simari

Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Artificial Intelligence Research and Development Laboratory (LIDIA)
Department of Computer Science and Engineering (DCIC)
Universidad Nacional del Sur (UNS), Bahía Blanca, Argentina
{ac,ajg,grs}@cs.uns.edu.ar

Abstract. This work will introduce a novel combination of two important argumentation related notions. We will start from the well-known basis of *Abstract Argumentation Frameworks* or *AFs*, and we will build a new formalism in which the notions corresponding to Toulmin's backings and Pollock's undercutting defeaters are considered. The resulting system, *Backing-Undercutting Argumentation Frameworks* or *BUAFs*, will be an extension of the *AFs* that includes a specialized support relation, a distinction between different attack types, and a preference relation among arguments. Thus, *BUAFs* will provide a richer representation tool for handling scenarios where information can be attacked and supported.

1 Introduction

Argumentation has been receiving increased attention as part of the Knowledge Representation and Reasoning area of Artificial Intelligence [4,18]. In short, argumentation is a form of reasoning where a piece of information (claim) is accepted or rejected after considering the reasons (arguments) for and against that acceptance. Thus, argumentation constitutes a reasoning mechanism with the capability of handling contradictory, incomplete and/or uncertain information. Several approaches were proposed to model argumentation: on an abstract basis [11], using classical logics [5], or using logic programming [12].

Argumentation models usually consider an argument as a piece of reasoning that provides a connection between some premises and a conclusion. Notwithstanding, in [19] Toulmin argued that arguments had to be analyzed using a richer format than the traditional one of formal logic. Whereas a formal logic analysis uses the dichotomy of premises and conclusion, Toulmin proposed a model for the layout of arguments that, in addition to data and claim, distinguishes four elements: warrant, backing, rebuttal and qualifier. However, Toulmin did not elaborate much on the nature of rebuttals, but simply stated that they provide conditions of exception for the argument. Therefore, without loss of generality, the notion of rebuttal can be paired to the notion of defeater for an argument, as proposed in the literature [17].

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 107–123, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

An important contribution to the field of argumentation which regards the nature of defeaters was proposed by Pollock. In [15] Pollock stated that defeasible reasons (which can be assembled to comprise arguments) have defeaters and that there are two kinds of defeaters: rebutting defeaters and undercutting defeaters. The former attack the conclusion of an inference by supporting the opposite one (*i. e.* they are reasons for denying the conclusion), while the latter attack the connection between the premises and conclusion without attacking the conclusion directly.

The contribution of this paper is to combine the notions presented by Toulmin and Pollock into an abstract argumentation framework. We will incorporate Pollock's categorization of defeaters and the modeling of Toulmin's scheme elements, in particular, focusing in undercutting defeaters and backings. We will follow the approach of [10] in which Pollock's undercutting defeaters can be regarded as attacking Toulmin's warrants. Thus, Toulmin's backings can be regarded as aiming to defend their associated warrants against undercutting attacks, by providing support for them. In that way, we will be able to capture both attack and support for inferences within the same context.

We will extend Abstract Argumentation Frameworks (AFs) [11] to incorporate a specialized type of support and preference relation among arguments, as well as distinguishing between different types of attacks. In particular, the support relation will correspond to the support that Toulmin's backings provide for their associated warrants. On the other hand, we will distinguish three different types of attack: rebutting attacks, undercutting attacks and undermining attacks; the first two being related to rebutting and undercutting defeaters, as proposed by Pollock. The remaining type of attack we will consider correspond to undermining defeaters, which are widely considered in the literature (see *e. g.* [16]) and originate from attacks to an argument's premise. We will also identify defeats that arise from the coexistence of backing and undercutting arguments, which will be shown to be conflicting. Later we will formalize properties regarding the characteristics of the framework and finally, following Dung's spirit, we will define the acceptability semantics for obtaining the sets of acceptable arguments of our framework.

The rest of this paper is organized as follows. Section 2 briefly reviews Dung's Abstract Argumentation Frameworks (AFs). In Section 3 we present the Backing-Undercutting Argumentation Frameworks (BUAFs), an extension of AFs that incorporates attack and support for inferences, as well as a preference relation to decide between conflicting arguments. In Section 4 we introduce the different types of defeat that can be obtained from a BUAF by applying preferences to the conflicting arguments. Later we define the requirements that a conflict-free set of arguments must satisfy. Section 5 introduces semantics-related notions, followed by the formal definitions of the acceptability semantics for BUAFs. Section 6 discusses related work and finally, in Section 7, some conclusions are commented.

oarieli@mta.ac.il

2 Dung's Abstract Argumentation Frameworks

In this section we will briefly review Dung's Abstract Argumentation Frameworks, as defined in [11].

Definition 1 ([11]). *An* Argumentation Framework (AF) *is a pair* $\langle \mathbb{A}, \mathbb{R} \rangle$, *where* $\mathbb{A}$ *is a set of arguments and* $\mathbb{R} \subseteq \mathbb{A} \times \mathbb{A}$ *is a defeat*[1] *relation.*

Here, arguments are abstract entities that will be denoted using calligraphic uppercase letters. No reference to the underlying logic is needed since the framework abstracts from the arguments structure. The defeat relation between two arguments $\mathcal{A}$ and $\mathcal{B}$ denotes the fact that these arguments cannot be accepted simultaneously since they are conflicting. An argument $\mathcal{A}$ *defeats* an argument $\mathcal{B}$ iff $(\mathcal{A}, \mathcal{B}) \in \mathbb{R}$, and it is noted as $\mathcal{A} \rightarrow \mathcal{B}$. For instance, in the AF of Figure 1 arguments $\mathcal{A}$ and $\mathcal{B}$ defeat each other, argument $\mathcal{B}$ defeats argument $\mathcal{C}$, and so on.

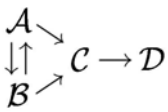


Fig. 1. A Dung's Abstract Argumentation Framework

Dung then defines the acceptability of arguments and the admissible sets of the framework.

Definition 2 ([11]). *Let* $AF = \langle \mathbb{A}, \mathbb{R} \rangle$ *be an argumentation framework and* $S \subseteq \mathbb{A}$ *a set of arguments. Then:*

- S *is* conflict-free *iff* $\nexists \mathcal{A}, \mathcal{B} \in S$ *s.t.* $(\mathcal{A}, \mathcal{B}) \in \mathbb{R}$.
- $\mathcal{A}$ *is* acceptable *w.r.t.* S *iff* $\forall \mathcal{B} \in \mathbb{A}$*: if* $(\mathcal{B}, \mathcal{A}) \in \mathbb{R}$ *then* $\exists \mathcal{C} \in S$ *s.t.* $(\mathcal{C}, \mathcal{B}) \in \mathbb{R}$.
- *If* S *is conflict-free, then* S *is an* admissible set *of* AF *iff each argument in* S *is acceptable w.r.t.* S.

Intuitively, an argument $\mathcal{A}$ is acceptable w.r.t. S if for any argument $\mathcal{B}$ that defeats $\mathcal{A}$, there is an argument $\mathcal{C}$ in S that defeats $\mathcal{B}$, in which case $\mathcal{C}$ is said to defend $\mathcal{A}$. An admissible set S can then be interpreted as a coherent defendable position. For instance, in the AF of Figure 1, argument $\mathcal{D}$ is acceptable w.r.t. the sets $\{\mathcal{A}\}$, $\{\mathcal{B}\}$ and $\{\mathcal{A}, \mathcal{B}\}$; however, only the first two of these sets are admissible.

Then, starting from the notion of admissibility, Dung defines the acceptability semantics of the framework.

[1] Dung originally uses the terminology '*attack*' in its definition; however, for the sake of clarity, we will rename Dung's attack relation to '*defeat*' relation.

oarieli@mta.ac.il

Definition 3 ([11]). *Let $AF = \langle \mathbb{A}, \mathbb{R} \rangle$ be an argumentation framework and $S \subseteq \mathbb{A}$ a conflict-free set of arguments. Then:*

- *S is a* complete extension *of AF iff all arguments acceptable w.r.t. S belong to S.*
- *S is a* preferred extension *of AF iff it is a maximal (w.r.t. set-inclusion) admissible set (i. e., a maximal complete extension).*
- *S is a* stable extension *of AF iff it defeats all arguments in $\mathbb{A} \backslash S$.*
- *S is the* grounded extension *of AF iff it is the smallest (w.r.t. set-inclusion) complete extension.*

The complete extensions of the framework in Figure 1 are $\emptyset$, $\{\mathcal{A}, \mathcal{D}\}$ and $\{\mathcal{B}, \mathcal{D}\}$; the preferred and stable extensions are $\{\mathcal{A}, \mathcal{D}\}$ and $\{\mathcal{B}, \mathcal{D}\}$; and the grounded extension is $\emptyset$.

3 Backing-Undercutting Argumentation Frameworks

A classical abstract argumentation framework is characterized by a set of arguments and a defeat relation among them. In this section, we will introduce an extension of Dung's argumentation frameworks, called Backing-Undercutting Argumentation Frameworks (BUAFs). In the extended framework we will distinguish between different types of attack, incorporate a specialized support relation, and include a preference relation to decide between conflicting arguments. Thus, BUAFs will provide the means for representing both attack and support for an argument's inference, allowing to capture Pollock's undercutting defeaters and Toulmin's backings.

Definition 4 (Backing-Undercutting Argumentation Framework). *A Backing-Undercutting Argumentation Framework (BUAF) is a tuple $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ where:*

- *$\mathbb{A}$ is a set of arguments,*
- *$\mathbb{D} \subseteq \mathbb{A} \times \mathbb{A}$ is an attack relation,*
- *$\mathbb{B} \subseteq \mathbb{A} \times \mathbb{A}$ is a backing relation, and*
- *$\preceq \subseteq \mathbb{A} \times \mathbb{A}$ is a partial order denoting a preference relation.*

We will distinguish three different types of attack within $\mathbb{D}$: the rebutting, undercutting and undermining attacks, respectively denoted as $\mathbb{R}_b$, $\mathbb{U}_c$, and $\mathbb{U}_m$ (*i. e.* $\mathbb{D} = \mathbb{R}_b \cup \mathbb{U}_c \cup \mathbb{U}_m$). In addition, a preference relation will be used to compare conflicting arguments in order to determine the successful attacks that result in defeats. Thus, when two arguments $\mathcal{A}$ and $\mathcal{B}$ are related by the preference relation (*i. e.* $(\mathcal{A}, \mathcal{B}) \in \preceq$) it means that argument $\mathcal{B}$ is at least as preferred as argument $\mathcal{A}$, denoting it as $\mathcal{A} \preceq \mathcal{B}$. As usual, $\mathcal{A} \prec \mathcal{B}$ means $\mathcal{A} \preceq \mathcal{B}$ and $\mathcal{B} \not\preceq \mathcal{A}$.

From hereon, we will use the following notation: $\mathcal{A} \dashrightarrow \mathcal{B}$ denotes $(\mathcal{A}, \mathcal{B}) \in \mathbb{D}$, and $\mathcal{A} \Longrightarrow \mathcal{B}$ denotes $(\mathcal{A}, \mathcal{B}) \in \mathbb{B}$. In order to illustrate, let us consider one

oarieli@mta.ac.il

of Toulmin's famous examples which discusses the nationality of a man named Harry [19], as shown in Figure 2.

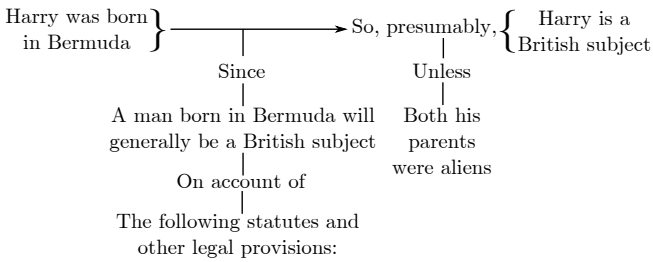


Fig. 2. Toulmin's example about Harry

The following arguments correspond to the situation described by Toulmin's example:

$\mathcal{H}$: *"Harry was born in Bermuda. A man born in Bermuda will generally be a British subject. So, presumably, Harry is a British subject"*

$\mathcal{B}$: *"On account of the following statutes and other legal provisions..."*

$\mathcal{U}$: *"Both Harry's parents are aliens"*

Example 1. *A possible representation for Toulmin's example about Harry is given by the BUAF* $\Delta_1 = \langle \mathbb{A}_1, \mathbb{D}_1, \mathbb{B}_1, \preceq_1 \rangle$*, where*

$$\mathbb{A}_1 = \{\mathcal{H},\ \mathcal{B},\ \mathcal{U}\} \qquad \mathbb{B}_1 = \{(\mathcal{B}, \mathcal{H})\}$$
$$\mathbb{U}_{c1} = \{(\mathcal{U}, \mathcal{H})\} \qquad \preceq_1 = \{(\mathcal{B}, \mathcal{U})\}$$

Here, that the existing statutes and other legal provisions provide support for the warrant of argument $\mathcal{H}$ *is expressed by the pair* $(\mathcal{B}, \mathcal{H})$ *in the backing relation. In addition, the fact that Harry's parents are aliens is an undercut for the inference, as expressed by the pair* $(\mathcal{U}, \mathcal{H})$ *in the attack relation.*

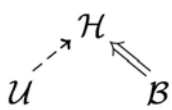


Fig. 3. The BUAF of Example 1

4 Defeat and Conflict-Freeness

Before defining any semantics-related notion, we must first consider the concept of defeat. Intuitively, given that in a BUAF preferences among arguments will be used to determine the success of attacks, an argument $\mathcal{A}$ would defeat an

oarieli@mta.ac.il

argument $\mathcal{B}$ iff $\mathcal{A}$ attacks $\mathcal{B}$ and $\mathcal{A}$ is not less preferred than $\mathcal{B}$. Following this intuition, in this section we will define the notion of defeat in the context of a BUAF, where we will introduce three types of defeat. Then, we will define a basic restriction that any acceptable set of arguments in a BUAF must satisfy, that is, the notion of conflict-freeness for a set of arguments.

The first type of defeat we will distinguish is called *primary defeat* and is obtained directly by resolving the attacks given on the attack relation through the use of preferences. In that way, primary defeats will always characterize the success of rebutting and undermining attacks. On the other hand, unlike other approaches (*e. g.* [16]), backings will be taken into consideration to determine the success of undercutting attacks. Hence, in the absence of backings, undercutting attacks will always succeed; otherwise, a further analysis will be required.

Definition 5 (Primary Defeat). *Let* $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ *be a BUAF and* $\mathcal{A}, \mathcal{B} \in \mathbb{A}$. *We will say that* $\mathcal{A}$ primary defeats $\mathcal{B}$ *iff one of the following conditions hold:*

- $(\mathcal{A}, \mathcal{B}) \in (\mathbb{R}_b \cup \mathbb{U}_m)$ *and* $\mathcal{A} \not\prec \mathcal{B}$, *or*
- $(\mathcal{A}, \mathcal{B}) \in \mathbb{U}_c$ *and* $\nexists \mathcal{C} \in \mathbb{A}$ *s.t.* $(\mathcal{C}, \mathcal{B}) \in \mathbb{B}$.

Remark 1. Observe that in the above definition rebutting and undermining attacks are grouped together. This is because, given the level of abstraction on the arguments structure, we can not distinguish an attack to an argument's premise from an attack to its conclusion. Thus, the only way to determine the existence of defeats in the presence of rebutting or undermining attacks is to compare the attacking and attacked arguments. In contrast, for instance, in concrete rule-based argumentation systems (*e. g.* [10]) this distinction between rebutting and undermining attacks becomes visible .

Example 2. *Continuing with Toulmin's scenario introduced on Example 1, suppose we add the following argument:*

> $\mathcal{P}$: *"Harry's birth certificate was found, and it states that Harry was born in Paris. So, Harry was not born in Bermuda."*

Argument $\mathcal{P}$ *undermines argument* $\mathcal{H}$*'s premise that Harry was born in Bermuda, originating an undermining attack from* $\mathcal{P}$ *to* $\mathcal{H}$. *In addition, suppose that the preference relation is extended to consider the new argument* $\mathcal{P}$, *and it is such that* $\mathcal{H} \preceq \mathcal{P}$. *The extended BUAF* $\Delta_2 = \langle \mathbb{A}_2, \mathbb{D}_2, \mathbb{B}_2, \preceq_2 \rangle$ *is included below:*

$$\begin{array}{ll} \mathbb{A}_2 = \{\mathcal{H},\ \mathcal{B},\ \mathcal{U},\ \mathcal{P}\} & \mathbb{B}_2 = \{(\mathcal{B}, \mathcal{H})\} \\ \mathbb{U}_{c2} = \{(\mathcal{U}, \mathcal{H})\} & \preceq_2 = \{(\mathcal{B}, \mathcal{U}), (\mathcal{H}, \mathcal{P})\} \\ \mathbb{U}_{m2} = \{(\mathcal{P}, \mathcal{H})\} & \end{array}$$

Here, argument $\mathcal{P}$ *primary defeats argument* $\mathcal{H}$ *given that the undermining attack succeeds.*

As stated before, in some cases, to determine whether an undercutting attack results in defeat it will be necessary to take backings into account. Following [10]'s

oarieli@mta.ac.il

approach, we will consider that backings are intended to defend their associated warrants against undercutting attacks. Therefore, it will be necessary to establish the relation between backing and undercutting arguments.

It is clear that backing and undercutting arguments are conflicting: while the latter attack the connection between premises and conclusion of an argument, the former provide support for it. Thus, they should not be jointly accepted. Moreover, given that the conflict between backing and undercutting arguments may not always be explicitly included on the attack relation of a BUAF, it is necessary to ensure this acceptability restriction. To achieve this, we will define a second type of defeat called *implicit defeat*.

Definition 6 (Implicit Defeat). *Let* $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ *be a BUAF and* $\mathcal{A}, \mathcal{B} \in \mathbb{A}$. *We will say that* $\mathcal{A}$ implicitly defeats $\mathcal{B}$ *iff one of the following conditions hold:*

- $(\mathcal{A}, \mathcal{C}) \in \mathbb{U}_c$ *and* $(\mathcal{B}, \mathcal{C}) \in \mathbb{B}$, *and* $\mathcal{A} \not\preceq \mathcal{B}$, *or*
- $(\mathcal{A}, \mathcal{C}) \in \mathbb{B}$ *and* $(\mathcal{B}, \mathcal{C}) \in \mathbb{U}_c$, *and* $\mathcal{A} \not\preceq \mathcal{B}$.

Example 3. *Given the BUAF of Example 2, argument* $\mathcal{U}$ *implicitly defeats argument* $\mathcal{B}$. *This is because argument* $\mathcal{U}$ *is an undercut for argument* $\mathcal{H}$, *whose backing is argument* $\mathcal{B}$, *and the preference relation is such that* $\mathcal{B} \preceq \mathcal{U}$.

Next, we will establish under what circumstances an attack in a BUAF succeeds. For that purpose, let us first consider the situation depicted by the BUAF $\Delta_3 = \langle \mathbb{A}_3, \mathbb{D}_3, \mathbb{B}_3, \preceq_3 \rangle$:

$$\mathbb{A}_3 = \{\mathcal{A}, \mathcal{B}, \mathcal{C}\} \qquad \mathbb{B}_3 = \{(\mathcal{B}, \mathcal{A})\}$$
$$\mathbb{R}_{b3} = \{(\mathcal{C}, \mathcal{B})\} \qquad \preceq_3 = \{(\mathcal{B}, \mathcal{C})\}$$

Here, argument $\mathcal{C}$ primary defeats argument $\mathcal{B}$ and, intuitively, the acceptable arguments from Δ_3 would be $\mathcal{C}$ and $\mathcal{A}$. However, recalling Toulmin's characterization of backings, a backing for an argument establishes the conditions why the connection between its premises and conclusion (*i. e.*, its associated warrant) holds. Therefore, in the above depicted situation, if the backing argument $\mathcal{B}$ is not acceptable, it implies that the conditions for argument $\mathcal{A}$'s warrant to hold are not satisfied. Thus, argument $\mathcal{A}$ should neither be acceptable since its associated warrant has no longer the necessary support, which was provided by argument $\mathcal{B}$. In order to prevent situations like this in a BUAF, we will introduce the *indirect defeats* among arguments.

Definition 7 (Indirect Defeat). *Let* $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ *be a BUAF and* $\mathcal{A}, \mathcal{B} \in \mathbb{A}$. *We will say that* $\mathcal{A}$ indirectly defeats $\mathcal{B}$ *iff* $\exists \mathcal{C} \in \mathbb{A}$ *s.t.* $(\mathcal{C}, \mathcal{B}) \in \mathbb{B}$ *and* $\mathcal{A}$ *primary defeats, implicitly defeats or indirectly defeats* $\mathcal{C}$.

The recursion in the preceding definition is necessary in order to capture the conflicts arising from a chaining of backing arguments, as shown in Figure 4. For instance, if we assume that argument $\mathcal{A}$ primary defeats argument $\mathcal{D}$, then we obtain an indirect defeat from $\mathcal{A}$ to $\mathcal{C}$. Furthermore, from this indirect defeat we also have that $\mathcal{A}$ indirectly defeats $\mathcal{B}$. This makes sense because if argument $\mathcal{C}$

oarieli@mta.ac.il

loses the support provided by its backing $\mathcal{D}$, then $\mathcal{C}$ has no longer the basis for providing the necessary support for argument $\mathcal{B}$.

$$\begin{array}{ccc} \mathcal{C} & \Longrightarrow & \mathcal{B} \\ \Uparrow & & \\ \mathcal{D} & \dashleftarrow & \mathcal{A} \end{array}$$

Fig. 4. Chaining of backing arguments

In particular, when backing arguments exist, undercutting defeats will be obtained by combining implicit and indirect defeats, as shown in the example below.

Example 4. *Continuing with Example 3, we know that argument $\mathcal{U}$ implicitly defeats argument $\mathcal{B}$. Hence, since argument $\mathcal{B}$ is a backing for argument $\mathcal{H}$, by Definition 7 we have that argument $\mathcal{U}$ indirectly defeats argument $\mathcal{H}$ and therefore, the undercutting attack from $\mathcal{U}$ to $\mathcal{H}$ is successful.*

Finally, we gather the different types of defeat within a single notion of defeat for BUAFs.

Definition 8 (Defeat). *Let $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ be a BUAF and $\mathcal{A}, \mathcal{B} \in \mathbb{A}$. Then $\mathcal{A}$* defeats *$\mathcal{B}$, noted as $\mathcal{A} \rightsquigarrow \mathcal{B}$, iff $\mathcal{A}$ primary defeats, implicitly defeats or indirectly defeats $\mathcal{B}$.*

From a BUAF Δ we can construct a directed graph called the *defeat graph*. The nodes in the graph are the arguments in Δ and the edges correspond to the defeat relation obtained by Definiton 8.

Example 5. *Suppose the following scenario where a group of friends is discussing about how long will it take to travel from city C_1 to city C_2 by car, given that the road distance between the cities is 300 km. During the discussion arguments $\mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{H}$ and $\mathcal{I}$ were exposed:*

> *$\mathcal{E}$: "We will drive at 120 km/h, and there is a highway from C_1 to C_2. Highways usually allow you to drive at constant speed. So, we should get to C_2 within 2:30 hours."*
>
> *$\mathcal{F}$: "Regulations state that the allowed max speed on highways is 120 km/h. Therefore, you can drive without stopping because that section of the highway is toll-free."*
>
> *$\mathcal{G}$: "I've heard on the news this morning that there was a car crash in that section of the highway. Thus, it was closed and the traffic was stopped."*
>
> *$\mathcal{H}$: "Yes, but that's old news. I've just heard on the radio that the highway was re-opened two hours ago."*

oarieli@mta.ac.il

$\mathcal{I}$: "Anyway, I've also heard that the highway got damaged after the accident. So, the max speed allowed within a 5 km radio from the crash site is 50 km/h."

Argument $\mathcal{F}$ provides support for argument $\mathcal{E}$'s warrant by establishing that since the highway is toll-free and the regulations allow it, is possible to drive at the desired speed. In contrast, argument $\mathcal{G}$ undercuts argument $\mathcal{E}$ by attacking the warrant that the highway allows driving at constant speed without stopping, since it was closed due to a car crash. On the other hand, argument $\mathcal{H}$ rebuts argument $\mathcal{G}$ by counter-arguing the conclusion that the highway is closed. Finally, argument $\mathcal{I}$ undermines argument $\mathcal{E}$'s premise of driving at 120 km/h, by stating that as a side effect from the accident the highway got damaged and the allowed maximum speed was reduced.

The above depicted situation can be characterized by the BUAF $\Delta_4 = \langle \mathbb{A}_4, \mathbb{D}_4, \mathbb{B}_4, \preceq_4 \rangle$:

$$\begin{array}{ll} \mathbb{A}_4 = \{\mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{H}, \mathcal{I}\} & \mathbb{U}_{m4} = \{(\mathcal{I}, \mathcal{E})\} \\ \mathbb{R}_{b4} = \{(\mathcal{H}, \mathcal{G})\} & \mathbb{B}_4 = \{(\mathcal{F}, \mathcal{E})\} \\ \mathbb{U}_{c4} = \{(\mathcal{G}, \mathcal{E})\} & \preceq_4 = \{(\mathcal{F}, \mathcal{G}), (\mathcal{G}, \mathcal{H}), (\mathcal{I}, \mathcal{E})\} \end{array}$$

Given the dynamics of the situation, the preference relation $\preceq_4$ prioritizes arguments with more recent information. Thus, we obtain the primary defeats $\mathcal{I} \rightsquigarrow \mathcal{E}$ and $\mathcal{H} \rightsquigarrow \mathcal{G}$, the implicit defeat $\mathcal{G} \rightsquigarrow \mathcal{F}$ and the indirect defeat $\mathcal{G} \rightsquigarrow \mathcal{E}$.

A graphical representation of Δ_4 and its corresponding defeat graph is shown in figures 5(a) and 5(b), respectively.

(a) (b)

Fig. 5. BUAF Δ_4 of Example 5 and its defeat graph

Given a BUAF $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ note that, by Definition 8, for any arguments $\mathcal{A}, \mathcal{B}, \mathcal{U} \in \mathbb{A}$ such that $(\mathcal{B}, \mathcal{A}) \in \mathbb{B}$ and $(\mathcal{U}, \mathcal{A}) \in \mathbb{U}_c$, it holds that:

- $\mathcal{B} \rightsquigarrow \mathcal{U}$ and $\mathcal{U} \not\rightsquigarrow \mathcal{B}$ iff $\mathcal{U} \prec \mathcal{B}$ (*i. e.* $\mathcal{U} \preceq \mathcal{B}$ and $\mathcal{B} \not\preceq \mathcal{U}$),
- $\mathcal{U} \rightsquigarrow \mathcal{B}$ and $\mathcal{B} \not\rightsquigarrow \mathcal{U}$ iff $\mathcal{B} \prec \mathcal{U}$ (*i. e.* $\mathcal{B} \preceq \mathcal{U}$ and $\mathcal{U} \not\preceq \mathcal{B}$), or
- $\mathcal{B} \rightsquigarrow \mathcal{U}$ and $\mathcal{U} \rightsquigarrow \mathcal{B}$ otherwise.

Next, conflict-free sets of arguments are characterized directly, by requiring the absence of defeats among the arguments belonging to the set.

oarieli@mta.ac.il

Definition 9 (Conflict-free Sets). *Let* $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ *be a BUAF. A set* $S \subseteq \mathbb{A}$ *is* conflict-free *iff* $\nexists \mathcal{A}, \mathcal{B} \in S$ *s.t.* $\mathcal{A} \rightsquigarrow \mathcal{B}$.

For instance, given the BUAF Δ_4 of Example 5, some conflict-free sets of arguments that we can distinguish are $\emptyset$, $\{\mathcal{E}\}$ and $\{\mathcal{F}, \mathcal{H}, \mathcal{I}\}$.

5 Acceptability Semantics

As introduced on Section 4, arguments in a BUAF may be conflicting and defeat each other and thus, they should not be jointly accepted. In order to do so, arguments in a BUAF will be subject to a status evaluation in which the accepted arguments will be those that somehow "survive" the defeats they receive. This evaluation process will be determined by the acceptability semantics of the framework.

In this section, we will define the basic semantic notions required for obtaining the sets of acceptable arguments of the framework. Then, we will formally define the acceptability semantics for BUAFs. Finally, a characterization of BUAFs as Dung's AFs is presented, establishing the relation between these two formalizations.

Definition 10 (Acceptability). *Let* $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ *be a BUAF. An argument* $\mathcal{A} \in \mathbb{A}$ *is* acceptable *w.r.t.* $S \subseteq \mathbb{A}$ *iff* $\forall \mathcal{B} \in \mathbb{A}$ *s.t.* $\mathcal{B} \rightsquigarrow \mathcal{A}$, $\exists \mathcal{C} \in S$ *s.t.* $\mathcal{C} \rightsquigarrow \mathcal{B}$.

Intuitively, an argument $\mathcal{A}$ will be acceptable with respect to a set of arguments S iff S defends $\mathcal{A}$ against all its defeaters. The following proposition shows that if an argument is defended by a set of arguments, then all its backings are also defended by that set.

Proposition 1. *Let* $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ *be a BUAF and* $\mathcal{A} \in \mathbb{A}$. *If* $\mathcal{A}$ *is acceptable w.r.t.* $S \subseteq \mathbb{A}$, *then:* $\forall \mathcal{B} \in \mathbb{A}$ *s.t.* $(\mathcal{B}, \mathcal{A}) \in \mathbb{B}, \mathcal{B}$ *is acceptable w.r.t.* S.

Proof. Let us suppose by contradiction that $\mathcal{A}$ *is acceptable w.r.t.* S *and* $\exists \mathcal{B} \in \mathbb{A}$ *s.t.* $(\mathcal{B}, \mathcal{A}) \in \mathbb{B}$ *and* $\mathcal{B}$ *is not acceptable w.r.t.* S. *Hence, by Definition 10,* $\exists \mathcal{C} \in \mathbb{A}$ *s.t.* $\mathcal{C} \rightsquigarrow \mathcal{B}$ *and* S *does not defend* $\mathcal{B}$ *against* $\mathcal{C}$. *Then, by Definition 7,* $\mathcal{C} \rightsquigarrow \mathcal{A}$. *Moreover, since* S *does not defend* $\mathcal{B}$ *against* $\mathcal{C}$, *it does not defend* $\mathcal{A}$ *either. Therefore, argument* $\mathcal{A}$ *would not be acceptable w.r.t.* S, *which contradicts our hypothesis.* □

Example 6. *Consider the BUAF* $\Delta_5 = \langle \mathbb{A}_5, \mathbb{D}_5, \mathbb{B}_5, \preceq_5 \rangle$, *where*

$$\begin{array}{ll} \mathbb{A}_5 = \{\mathcal{J}, \mathcal{K}, \mathcal{L}, \mathcal{M}, \mathcal{N}, \mathcal{O}, \mathcal{P}\} & \mathbb{U}_{m5} = \{(\mathcal{N}, \mathcal{M})\} \\ \mathbb{R}_{b5} = \{(\mathcal{J}, \mathcal{K})\} & \mathbb{B}_5 = \{(\mathcal{L}, \mathcal{K}), (\mathcal{P}, \mathcal{N})\} \\ \mathbb{U}_{c5} = \{(\mathcal{M}, \mathcal{K}), (\mathcal{O}, \mathcal{N})\} & \preceq_5 = \{(\mathcal{J}, \mathcal{K}), (\mathcal{L}, \mathcal{M}), (\mathcal{M}, \mathcal{N})\} \end{array}$$

The defeat graph for Δ_5 *is*

$$\begin{tikzcd} \mathcal{J} & \mathcal{K} & & \mathcal{P} \\ & \mathcal{M} \arrow[u, leftrightsquigarrow] & & \\ \mathcal{L} \arrow[r, leftrightsquigarrow] & \mathcal{M} \arrow[r, leftrightsquigarrow] & \mathcal{N} \arrow[r, leftrightsquigarrow] & \mathcal{O} \arrow[uu, leftrightsquigarrow] \end{tikzcd}$$

oarieli@mta.ac.il

Here, argument $\mathcal{K}$ is acceptable w.r.t. the sets $\{\mathcal{N}\}$, $\{\mathcal{N}, \mathcal{P}\}$ and $\{\mathcal{N}, \mathcal{J}, \mathcal{O}\}$ among others. Note that the backing $\mathcal{L}$ for argument $\mathcal{K}$ is also acceptable w.r.t. those sets, as stated by Proposition 1.

A usual requirement when defining the set of acceptable arguments of an argumentation framework is the conflict-freeness of the set (see *e. g.*, [11,3]). This implies that a set of collectively acceptable arguments must be internally coherent, in the sense that no pair of arguments belonging to the set defeat each other. On the other hand, given that BUAFs incorporate support among arguments through the backing relation, an acceptable set of arguments from a BUAF must also satisfy some *external coherence*. Thus, no pair of arguments within the set of accepted arguments must be implicitly conflicting.

Intuitively, a set of arguments will be externally coherent if no pair of arguments in the set simultaneously defeat and support another argument. However, due to the nature of the support relation being modeled by BUAFs (which represents the support provided by Toulmin's backings for their warrants), we would only consider undercutting defeats as threats to the external coherence of a set of arguments.

Definition 11 (External Coherence). *Let $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ be a BUAF. A set $S \subseteq \mathbb{A}$ is* externally coherent *iff $\forall \mathcal{C} \in \mathbb{A} : \nexists\ \mathcal{A}, \mathcal{B} \in S$ s.t. $(\mathcal{A}, \mathcal{C}) \in \mathbb{U}_c$, $\mathcal{A} \rightsquigarrow \mathcal{C}$, and $(\mathcal{B}, \mathcal{C}) \in \mathbb{B}$.*

Example 7. *Given the BUAF Δ_5 of Example 6, for instance, the sets $\{\mathcal{J}, \mathcal{K}, \mathcal{L}\}$ and $\{\mathcal{M}, \mathcal{N}\}$ are externally coherent, while the set $\{\mathcal{O}, \mathcal{P}\}$ is not.*

The following proposition shows that conflict-freenes suffices to assure external coherence.

Proposition 2. *Let $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ be a BUAF and $S \subseteq \mathbb{A}$. If S is conflict-free, then S is externally coherent.*

Proof. Let us assume by contradiction that S is conflict-free and not externally coherent. Then, by Definition 11, $\exists \mathcal{A}, \mathcal{B} \in S$, $\exists \mathcal{C} \in \mathbb{A}$ s.t. $(\mathcal{A}, \mathcal{C}) \in \mathbb{U}_c$, $\mathcal{A} \rightsquigarrow \mathcal{C}$, and $(\mathcal{B}, \mathcal{C}) \in \mathbb{B}$. This entails by Definition 6 that either $\mathcal{A} \rightsquigarrow \mathcal{B}$, $\mathcal{B} \rightsquigarrow \mathcal{A}$, or $\mathcal{A} \rightsquigarrow \mathcal{B}$ and $\mathcal{B} \rightsquigarrow \mathcal{A}$, which contradicts the hypothesis that S is conflict-free. □

In particular, when $\mathcal{A} = \mathcal{B}$ in Definition 11, the characteristic of external coherence becomes into *consistency*, which clearly is an essential requirement for any set of acceptable arguments. This is because an argument that simultaneously supports and defeats another argument is an inconsistent piece of reasoning and therefore, it should be disregarded when obtaining the set of acceptable arguments of the framework.

Definition 12 (Consistency). *Let $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ be a BUAF. A set $S \subseteq \mathbb{A}$ is* consistent *iff $\forall \mathcal{B} \in \mathbb{A} : \nexists \mathcal{A} \in S$ s.t. $(\mathcal{A}, \mathcal{B}) \in \mathbb{U}_c$, $\mathcal{A} \rightsquigarrow \mathcal{B}$, and $(\mathcal{A}, \mathcal{B}) \in \mathbb{B}$.*

Example 8. *Continuing with Example 7, the sets $\{\mathcal{J}, \mathcal{K}, \mathcal{L}\}$, $\{\mathcal{M}, \mathcal{N}\}$ and $\{\mathcal{O}, \mathcal{P}\}$ are consistent.*

oarieli@mta.ac.il

The following proposition shows that, effectively, consistency is a particular case of external coherence.

Proposition 3. *Let* $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ *be a BUAF and* $S \subseteq \mathbb{A}$*. If* S *is externally coherent, then* S *is consistent.*

Proof. Let us assume by contradiction that S *is externally coherent and not consistent. Then, by Definition 12,* $\exists \mathcal{A} \in S$, $\exists \mathcal{C} \in \mathbb{A}$ *s.t.* $(\mathcal{A}, \mathcal{C}) \in \mathbb{U}_c$, $\mathcal{A} \rightsquigarrow \mathcal{C}$, *and* $(\mathcal{A}, \mathcal{C}) \in \mathbb{B}$*. This entails by Definition 11 that* $\exists \mathcal{A}, \mathcal{B} \in S$, $\exists \mathcal{C} \in \mathbb{A}$ *s.t.* $\mathcal{B} = \mathcal{A}$, $(\mathcal{A}, \mathcal{C}) \in \mathbb{U}_c$, $\mathcal{A} \rightsquigarrow \mathcal{C}$, *and* $(\mathcal{B}, \mathcal{C}) \in \mathbb{B}$*. Therefore,* S *is not externally coherent, which contradicts the hypothesis.* □

The previously mentioned characteristics (external coherence and consistency) represent desirable features for any set of acceptable arguments of a BUAF. Thus, by propositions 2 and 3, conflict-freeness suffices to assure external coherence and consistency. That the reverse does not hold is shown by Example 9.

Example 9. *Given the BUAF* Δ_5 *of Example 6, it was shown that the set* $\{\mathcal{M}, \mathcal{N}\}$ *is externally coherent and consistent; however it is not conflict-free since there is an undermining defeat from* $\mathcal{N}$ *to* $\mathcal{M}$*.*

We have proved that a conflict-free set of arguments satisfies the desired features of external coherence and consistency, which are characteristics that any acceptable set of arguments should satisfy. Hence, we are able to define the notion of admissibility similarly to [11] without requiring additional constraints.

Definition 13 (Admissibility). *Let* $\langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ *be a BUAF. A set* $S \subseteq \mathbb{A}$ *is* admissible *iff it is conflict-free and all elements of* S *are acceptable w.r.t.* S*.*

Example 10. *From the sets of arguments listed in Example 6, the only admissible set is* $\{\mathcal{N}, \mathcal{P}\}$*. The set* $\{\mathcal{N}\}$ *is not admissible since it does not defend* $\mathcal{N}$ *against* $\mathcal{O}$*. On the other hand, the set* $\{\mathcal{N}, \mathcal{J}, \mathcal{O}\}$ *is not admissible given that it is not conflict-free.*

Recall that acceptability semantics identify a set of extensions of an argumentation framework, namely sets of arguments which are collectively acceptable. The complete, preferred, stable and grounded extensions of a BUAF are now defined in the same way as for Dung's frameworks.

Definition 14 (Extensions). *Let* $\Delta = \langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ *be a BUAF and* $S \subseteq \mathbb{A}$ *a conflict-free set of arguments. Then:*

- S *is a* complete extension *of* Δ *iff all arguments acceptable w.r.t.* S *belong to* S*.*
- S *is a* preferred extension *of* Δ *iff it is a maximal (w.r.t. set-inclusion) admissible set of* Δ *(i. e., a maximal complete extension).*
- S *is a* stable extension *of* Δ *iff it defeats all arguments in* $\mathbb{A} \backslash S$*.*
- S *is the* grounded extension *of* Δ *iff it is the smallest (w.r.t. set-inclusion) complete extension.*

oarieli@mta.ac.il

Given a BUAF and a semantics s, an argument $\mathcal{A}$ will be *skeptically accepted* if it belongs to all s-extensions; $\mathcal{A}$ will be *credulously accepted* if it belongs to some (not all) s-extensions; and $\mathcal{A}$ will be *rejected* otherwise.

Example 11. *From the BUAF Δ_5 of Example 6 we can obtain the following sets of extensions:*

- *the complete extensions $\{\mathcal{J}\}$, $\{\mathcal{J},\mathcal{K}\}$, $\{\mathcal{J},\mathcal{L}\}$, $\{\mathcal{J},\mathcal{K},\mathcal{L}\}$, $\{\mathcal{J},\mathcal{K},\mathcal{L},\mathcal{N}\}$, $\{\mathcal{J},\mathcal{K},\mathcal{L},\mathcal{N},\mathcal{P}\}$, $\{\mathcal{J},\mathcal{M}\}$ and $\{\mathcal{J},\mathcal{M},\mathcal{O}\}$;*
- *the preferred and stable extensions $\{\mathcal{J},\mathcal{K},\mathcal{L},\mathcal{N},\mathcal{P}\}$ and $\{\mathcal{J},\mathcal{M},\mathcal{O}\}$; and*
- *the grounded extension $\{\mathcal{J}\}$.*

Definitions 10, 13 and 14 are analogous to those presented for Dung's argumentation frameworks. Recall that a classical argumentation framework is characterized by a set of arguments and a defeat relation among them. Thus, using the defeat relation from Definition 8 and the set of arguments of a BUAF we can characterize an abstract argumentation framework which accepts exactly the same arguments as the BUAF under the same semantics.

Proposition 4. *Let $\Delta = \langle \mathbb{A}, \mathbb{D}, \mathbb{B}, \preceq \rangle$ be a BUAF. There exists an abstract argumentation framework $AF = \langle \mathbb{A}, \rightsquigarrow$ [2]$\rangle$ such that the sets of extensions of Δ and AF under a given semantics are equal.*

Proof. Straightforward from definitions 2, 3, 10, 13 and 14. □

By Proposition 4, BUAFs will inherit all properties from abstract argumentation frameworks (refer to [11] for details). Moreover, it will be possible to determine the acceptability of arguments in a BUAF using its associated AF. We first obtain the associated AF and then, Dung's acceptability semantics are applied to this AF.

6 Related Work

We have presented an extension of Dung's AFs that enables the representation of Toulmin's backings and Pollock's undercutting defeaters. Although several approaches address these two notions separately, they were not widely considered together in the formalizations provided so far. For instance, in [16] an extension of AFs is presented, where arguments are partly provided of an internal structure and a categorization of defeaters is also given, allowing to model undercutting defeaters; however, in that work there is no consideration for support among arguments.

Likewise [1], our approach incorporates a preference relation among arguments in order to determine the success of attacks. In addition, in [2] the authors show that preferences pay two roles in argumentation frameworks: i) to compute standard solutions (*i. e.*, extensions), and ii) to refine those solutions (*i. e.*, to return

[2] The defeat relation given in Definition 8.

oarieli@mta.ac.il

only the preferred extensions). Other works that consider preferences among arguments include [13] and [3], but the difference between those approaches and ours is that they express preferences at the object level, by incorporating attacks to attacks. That is, they incorporate a high-level attack relation from arguments to attacks, where a pair ($\mathcal{C}$, ($\mathcal{A}, \mathcal{B}$)) in that relation can be interpreted as "$\mathcal{C}$ claims that $\mathcal{B}$ is preferred to $\mathcal{A}$".

A formalization that addresses support among arguments correspond to the Bipolar Argumentation Frameworks (BAFs) [7]. A Bipolar Argumentation Framework extends Dung's AF to incorporate a support relation among arguments. Then, the authors identify new attacks originated from the conflicts between supporting and attacking arguments. In addition, the authors defined some extra requirements for admissible sets of arguments, such as external coherence and consistency. Finally, the acceptable sets of arguments of a BAF are characterized in two different ways: the former by defining acceptability semantics that take the new conflicts into account, and the latter by grouping arguments into coalitions that are afterwards considered as a whole to compute the extensions.

The main difference between BAFs and BUAFs is that the support relation in a BAF is general, while the backing relation proposed in this work corresponds to the support relation between Toulmin's backings and warrants. Therefore, the implicit conflicts arising from backing and undercutting arguments could not be captured by BAFs. In addition, we have shown that the constraints of external coherence and consistency presented for BAFs in [7] are also satisfied by the notion of admissibility given in our proposal.

Another approach to abstract argumentation frameworks that takes support among arguments into account is presented in [14]. There, the authors propose the Argumentation Frameworks with Necessities (AFNs) in which two interpretations for the support relation are given: the necessity and the sufficiency relations. These relations are interpreted by considering "$\mathcal{A}$ *is necessary for* $\mathcal{B}$" exactly as "$\mathcal{B}$ *is sufficient for* $\mathcal{A}$", meaning that "*if* $\mathcal{B}$ *is accepted then* $\mathcal{A}$ *is accepted*". Given the provided interpretations, the authors pose the duality between the two relations, allowing them to focus only in the necessity relation. Several conflicts arising from attacking and supporting arguments are detected, and then the corresponding acceptability semantics are defined. In addition, the authors show how the necessity relation allows for a correspondence between a fragment of logic programs and AFNs. Finally, they introduce a generalization of AFNs that extends the necessity relation to deal with sets of arguments.

On the other hand, a Meta-Argumentation approach that takes support among arguments into account was presented in [6]. In that work, the support relation is considered as deductive support. Thus, "$\mathcal{A}$ *supports* $\mathcal{B}$" is interpreted as "*if* $\mathcal{A}$ *is accepted then* $\mathcal{B}$ *is accepted*" and, as a consequence, "if $\mathcal{B}$ is not accepted then $\mathcal{A}$ is not accepted". Besides capturing the attacks originated from the combination of attacks and supports, the authors introduce defeasible support by stating that the implication associated to the deductive support holds by default and can be attacked. Thus, to capture this intuition they introduce second-order attacks from an argument to the support relation.

oarieli@mta.ac.il

A comprehensive comparison among the above mentioned formalisms was given by [8]. In that work the authors remark that although deductive support (*d-support*) and necessary support (*n-support*) have been introduced independently in [6] and [14] respectively, they correspond to dual interpretations of the support relation in the following sense: "$\mathcal{A}$ *n-supports* $\mathcal{B}$" is equivalent to "$\mathcal{B}$ *d-supports* $\mathcal{A}$". Thus, by inverting the direction of deductive support in (which gives necessity relations), the attacks defined by the authors in [6] correspond respectively to the extended attacks proposed in [14]. Notwithstanding, due to the variety on the nature of the support relation being modeled, none of these approaches can capture the conflicts arising from the coexistence of backing and undercutting arguments. That is to say, the implicit defeats originated from arguments supporting and respectively attacking an inference.

In [21] Verheij reconstructed Toulmin's ideas using a theory of dialectical argumentation called DefLog [20]. Briefly, its logical language has two connectives $\times$ and $\sim>$. The dialectical negation $\times S$ of a statement S expresses that the statement S is defeated. The primitive implication $\sim>$ is a binary connective used to express that one statement supports another, and only validates modus ponens. In DefLog is possible to combine and nest the connectives $\times$ and $\sim>$ to obtain more complex statements, allowing to represent both Toulmin's backings and Pollock's undercutting defeaters. Nevertheless, since dialectical negation indicates defeat, an argument for a statement $\times S$ will always be preferred to an argument for a statement S. Thus, in Verheij's approach it is not possible to express attack without defeat. On the contrary, attacks in a BUAF do not always result in defeat. Moreover, for determining the success of undercutting attacks in a BUAF, the existence of backings needs to be taken into consideration.

7 Conclusions

In this work, an extension of abstract argumentation frameworks called Backing-Undercutting Argumentation Frameworks (BUAFs) was proposed, inspired by the work of Toulmin [19] and Pollock [15]. This extension allows to model scenarios where attack and support for inferences may appear, by distinguishing different types of attacks and incorporating a specialized support relation among arguments. In that way, the extended framework enables the representation of Toulmin's backings and Pollock's undercutting defeaters, two important notions within the argumentation community.

Several approaches address these two notions separately, yet they were not widely considered together in the formalizations provided so far. That is, although the existing works that address support in argumentation frameworks also take attacks into account, there is much to study about the possible conflicts arising from the coexistence of attacking and supporting arguments. In particular, the current approaches can not capture the implicit conflicts arising from the combination of backings and undercutting defeaters.

Finally, it was shown that BUAFs can be mapped to AFs by considering the set of arguments and the corresponding defeat relation. Thus, it is clear

oarieli@mta.ac.il

that the examples and applications shown for BUAFs can also be modeled with Dung's abstract frameworks. Notwithstanding this observation, it is important to remark that in addition to formalizing the backing relation and different types of attack, BUAFs provide a more specific and intuitive tool for representing argumentative or nonmonotonic scenarios where information may be attacked and supported. This work has served to further the research on the possible extensions of abstract argumentation frameworks using existing research in the area of concrete argumentation introduced in [19,15] and already incorporated in existing implementations [9,10].

References

1. Amgoud, L., Cayrol, C.: A reasoning model based on the production of acceptable arguments. Annals of Mathematics and Artificial Intelligence 34(1-3), 197–215 (2002)
2. Amgoud, L., Vesic, S.: Two Roles of Preferences in Argumentation Frameworks. In: Liu, W. (ed.) ECSQARU 2011. LNCS, vol. 6717, pp. 86–97. Springer, Heidelberg (2011)
3. Baroni, P., Cerutti, F., Giacomin, M., Guida, G.: AFRA: Argumentation Framework with Recursive Attacks. Int. Journal of Approximate Reasoning 52(1), 19–37 (2011)
4. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artificial Intelligence 171(10-15), 619–641 (2007)
5. Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. Artificial Intelligence 128(1-2), 203–235 (2001)
6. Boella, G., Gabbay, D.M., van der Torre, L.W.N., Villata, S.: Support in abstract argumentation. In: Baroni, P., Cerutti, F., Giacomin, M., Simari, G.R. (eds.) 3rd Int. Conf. on Computational Models of Argument (COMMA 2010). Frontiers in Artificial Intelligence and Applications, vol. 216, pp. 111–122. IOS Press (2010)
7. Cayrol, C., Lagasquie-Schiex, M.-C.: Bipolar abstract argumentation systems. In: Simari, G.R., Rahwan, I. (eds.) Argumentation in Artificial Intelligence, pp. 65–84. Springer, US (2009)
8. Cayrol, C., Lagasquie-Schiex, M.-C.: Bipolarity in Argumentation Graphs: Towards a Better Understanding. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 137–148. Springer, Heidelberg (2011)
9. Cohen, A., García, A.J., Simari, G.R.: Extending DeLP with Attack and Support for Defeasible Rules. In: Kuri-Morales, A., Simari, G.R. (eds.) IBERAMIA 2010. LNCS, vol. 6433, pp. 90–99. Springer, Heidelberg (2010)
10. Cohen, A., García, A.J., Simari, G.R.: Backing and Undercutting in Defeasible Logic Programming. In: Liu, W. (ed.) ECSQARU 2011. LNCS, vol. 6717, pp. 50–61. Springer, Heidelberg (2011)
11. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–358 (1995)
12. García, A.J., Simari, G.R.: Defeasible logic programming: An argumentative approach. Theory and Practice of Logic Programming (TPLP) 4(1-2), 95–138 (2004)
13. Modgil, S.: Reasoning about preferences in argumentation frameworks. Artificial Intelligence 173(9-10), 901–934 (2009)

oarieli@mta.ac.il

14. Nouioua, F., Risch, V.: Argumentation Frameworks with Necessities. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 163–176. Springer, Heidelberg (2011)
15. Pollock, J.L.: Defeasible reasoning. Cognitive Science 11(4), 481–518 (1987)
16. Prakken, H.: An abstract framework for argumentation with structured arguments. Journal of Argument and Computation 1, 93–124 (2009)
17. Prakken, H., Vreeswijk, G.: Logics for defeasible argumentation. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. 4, pp. 218–319. Kluwer Academic Pub. (2002)
18. Rahwan, I., Simari, G.R.: Argumentation in Artificial Intelligence. Springer, Heidelberg (2009)
19. Toulmin, S.E.: The Uses of Argument. Cambridge University Press (1958)
20. Verheij, B.: DefLog: On the logical interpretation of prima facie justified assumptions. Journal of Logic and Computation 13(3), 319–346 (2003)
21. Verheij, B.: Evaluating arguments based on Toulmin's scheme. Argumentation 19(3), 347–371 (2005)

oarieli@mta.ac.il

The Impact of Transitive Closure on the Boolean Expressiveness of Navigational Query Languages on Graphs

George H.L. Fletcher[1], Marc Gyssens[2], Dirk Leinders[2,⋆], Jan Van den Bussche[2], Dirk Van Gucht[3], Stijn Vansummeren[4], and Yuqing Wu[3]

[1] Eindhoven University of Technology
g.h.l.fletcher@tue.nl
[2] Hasselt University and Transnational University of Limburg
School for Information Technology
{marc.gyssens,dirk.leinders,jan.vandenbussche}@uhasselt.be
[3] Indiana University
{vgucht,yuqwu}@cs.indiana.edu
[4] Université Libre de Bruxelles
stijn.vansummeren@ulb.ac.be

Abstract. Several established and novel applications motivate us to study the expressive power of navigational query languages on graphs, which represent binary relations. Our basic language has only the operators union and composition, together with the identity relation. Richer languages can be obtained by adding other features such as other set operators, projection and coprojection, converse, and the diversity relation. In this paper, we show that, when evaluated at the level of boolean queries with an unlabeled input graph (i.e., a single relation), adding transitive closure to the languages with coprojection adds expressive power, while this is *not* the case for the basic language to which none, one, or both of projection and the diversity relation are added. In combination with earlier work [10], these results yield a complete understanding of the impact of transitive closure on the languages under consideration.

1 Introduction

In previous work [10], the present authors studied the relative expressive power of query languages on graphs (i.e., binary relations). They considered a basic language, consisting of union, composition, and the identity relation, to which one or more features can be added, such as intersection, set difference, projection, coprojection, converse, and the diversity relation. We refer to the basic language to which all the non-basic features have been added as the *relation algebra*.

A relation algebra expression can be seen as a function mapping the input binary relation to a binary relation. We call such queries *path queries* because the result can be interpreted as all the ways in which the input graph can be

⋆ This author carried out most of his research as a Senior Research Assistant of the Fund of Scientific Research – FWO Flanders.

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 124–143, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

navigated in accordance with the expression. By identifying nonemptiness with the boolean value *true* and emptiness with *false*, as is standard in database theory [3], we can also express yes/no queries within this framework. To distinguish them from general path queries, we shall refer to the latter as *boolean queries*.

The present authors were able to establish the complete Hasse diagram for the relative expressive power of the various relation algebra fragments, and this both at the levels of (1) path queries and (2) boolean queries, both for the cases where the input graph is (1) labeled (i.e., may represent multiple binary relations) and (2) is unlabeled (i.e., represents a single relation).

This study was motivated by similar work on the expressive power of XPath fragments as query languages for navigating on trees, which is now well understood (e.g., [7,13,20,21,25]). Motivated by data on the Web [2,11] and new applications such as dataspaces [12], Linked Data [8,15], and RDF [1], it is natural to look at similar navigational query languages for graphs. The languages we study are very natural and similar to languages already considered in the fields of description logics, dynamic logics, arrow logics, and relation algebras [6,9,14,17,19,23]. Moreover, graph query languages have a rich history in database theory, in particular in the context of object-oriented and semi-structured database systems. We refer to Angles and Gutiérrez [5] for a comprehensive review.

In addition to what has been described above, we also investigated whether adding transitive closure to a relation algebra fragment yields additional expressive power. At the level of path queries, this is obviously the case for all fragments, as the transitive closure of a binary relation is not expressible in FO [4], whereas the full relation algebra is known to be equivalent to FO^3 [24]. We were also able to show [10] that adding transitive closure does not result in a collapse at the level of boolean queries, provided the input graph is labeled (i.e., there may be several input relations). For boolean queries on unlabeled graphs (i.e., on a single input relation), several cases remained open, however.

The purpose of the present paper is to solve these cases, and thus to complete our understanding of whether or not the relation algebra fragments with transitive closure collapse to their counterparts without transitive closure at the level of boolean queries on unlabeled graphs.

To see the practical relevance of these results, consider the following example. Facebook is a large social network which maintains a graph of people that are connected via a friendship relationship. It is customary that people wish to communicate with their friends, navigate recursively to friends of friends etc. This navigation can be expressed with path expressions in a suitable relation algebra fragment, either with or without using transitive closure. In addition to navigation, certain topological properties of the Facebook graph can be discovered. For example, one can discover whether there are people whose friends are all friends of each other. Again, some of these topological properties can be formulated as boolean queries in a suitably chosen relation algebra fragment, either with or without using transitive closure. The proliferation of social networks is thus a real-world phenomenon to which our theory applies.

oarieli@mta.ac.il

From this perspective, the collapse results are very meaningful.

With regard to the possible collapse of the various relation algebra fragments with transitive closure to their counterparts without transitive closure at the level of boolean queries on unlabeled graphs, it was already established [10] that adding transitive closure to a relation algebra fragment adds expressive power if either intersection, or converse, or both, can be expressed in that fragment. In particular, this is the case when set difference is in the fragment.

It thus remained to look at relation algebra fragments consisting of the basic language built from union, composition, and the identity relation, to which a subset of the features projection, coprojection, and the diversity relation has been added. It was also established [10] that adding transitive closure to (1) the basic language, (2) the basic language to which projection is added, and (3) the basic language to which the diversity relation is added does *not* increase the expressive power.

Taking into account that projection can be expressed as the coprojection of the coprojection, three cases remained open, however. Does adding transitive closure to (1) the basic language to which coprojection is added, (2) the basic language to which both coprojection and the diversity relation are added, and (3) the basic language to which both projection and the diversity relation are added increase the expressive power?

In this paper, we show that there is no collapse for the first two fragments, but that there is a collapse for the third fragment. The emphasis of this paper is on the proof technique used for establishing the collapse for this last fragment, which we think is interesting in its own right.

The paper is organized as follows. In Section 2, we define syntax and semantics of the class of languages studied in the paper. In Section 3, we show that for any relation algebra fragment in which coprojection can be expressed, adding transitive closure yields additional expressive power at the level of boolean queries, thus settling the previously open cases for (1) the basic language to which coprojection is added and (2) the basic language to which both coprojection and the diversity relation are added. In Section 4, we describe a two-step proof strategy to show that adding transitive closure to (3) the basic language to which both projection and the diversity relation are added does not increase the expressive power, and we deal with the first step. In Sections 5 to 8, we deal with the much more elaborate second step. We conclude in Section 9 by summarizing our understanding of the impact of adding transitive closure to relation algebra fragments, which has now been completed.

Finally, notice that, in this extended abstract, most proofs have either been omitted or only summarily sketched.

2 Graphs and Languages

In this paper, we are interested in navigating over graphs. For our purposes, a graph is a relational structure G, consisting of a set of nodes V and a binary relation $R \subseteq V \times V$, the set of edges of G. In what follows, both V and R may be either finite or infinite.

oarieli@mta.ac.il

An extension of this model consists of allowing multiple binary relations, by labeling the edges.[1] For comparison, we shall sometimes refer to labeled graphs, though the emphasis of this paper is on unlabeled graphs.

The most basic language for navigating over graphs we consider is the algebra $\mathcal{N}$ whose expressions are built recursively from the edge set symbol R, the primitive $\emptyset$, and the primitive id, using composition $(e_1 \circ e_2)$ and union $(e_1 \cup e_2)$.[2]

Semantically, each expression $e \in \mathcal{N}$ defines a path query. A path query takes as input a graph G and returns a binary relation $e(G) \subseteq \mathrm{adom}(G) \times \mathrm{adom}(G)$, where $\mathrm{adom}(G)$ denotes the *active domain* of G, which is the set of all entries occurring in one of the relations of G, i.e.,

$$\mathrm{adom}(G) = \{v \mid \exists w : (v, w) \in R \vee (w, v) \in R\}.$$

In particular, the semantics of $\mathcal{N}$ is inductively defined as follows:

$$\begin{aligned}
R(G) &= R\,; \\
\emptyset(G) &= \emptyset\,; \\
id(G) &= \{(v, v) \mid v \in \mathrm{adom}(G)\}\,; \\
e_1 \circ e_2(G) &= \{(v, w) \mid \exists z : (v, z) \in e_1(G) \ \&\ (z, w) \in e_2(G))\}\,; \\
e_1 \cup e_2(G) &= e_1(G) \cup e_2(G)\,.
\end{aligned}$$

The basic algebra $\mathcal{N}$ can be extended by adding some of the following features: diversity (di), converse (e^{-1}), intersection ($e_1 \cap e_2$), difference ($e_1 \setminus e_2$), projections ($\pi_1(e)$ and $\pi_2(e)$), coprojections ($\overline{\pi}_1(e)$ and $\overline{\pi}_2(e)$), and transitive closure (e^+). We refer to the operators in the basic algebra $\mathcal{N}$ as *basic features*; we refer to the extensions as *nonbasic features*. The semantics of the extensions is as follows:

$$\begin{aligned}
di(G) &= \{(v, w) \mid v, w \in \mathrm{adom}(G) \ \&\ v \neq w\}\,; \\
e^{-1}(G) &= \{(v, w) \mid (w, v) \in e(G)\}\,; \\
e_1 \cap e_2(G) &= e_1(G) \cap e_2(G)\,; \\
e_1 \setminus e_2(G) &= e_1(G) \setminus e_2(G)\,; \\
\pi_1(e)(G) &= \{(v, v) \mid v \in \mathrm{adom}(G) \ \&\ \exists w : (v, w) \in e(G)\}\,; \\
\pi_2(e)(G) &= \{(v, v) \mid v \in \mathrm{adom}(G) \ \&\ \exists w : (w, v) \in e(G)\}\,; \\
\overline{\pi}_1(e)(G) &= \{(v, v) \mid v \in \mathrm{adom}(G) \ \&\ \neg\exists w : (v, w) \in e(G)\}\,; \\
\overline{\pi}_2(e)(G) &= \{(v, v) \mid v \in \mathrm{adom}(G) \ \&\ \neg\exists w : (w, v) \in e(G)\}\,; \\
e^+(G) &= \textstyle\bigcup_{k \geq 1} e^k(G)\,.
\end{aligned}$$

Here, e^k denotes $e \circ \cdots \circ e$ (k times).

If F is a set of nonbasic features, we denote by $\mathcal{N}(F)$ the language obtained by adding all features in F to $\mathcal{N}$. For example, $\mathcal{N}(\cap)$ denotes the extension of $\mathcal{N}$ with intersection, and $\mathcal{N}(\cap, \pi, +)$ denotes the extension of $\mathcal{N}$ with intersection, both projections,[3] and transitive closure.

[1] In this case, the number of relation names is always finite.

[2] By abuse of notation, we shall use "R" both as a symbol in the algebra $\mathcal{N}$ and as the name of the corresponding edge relation in G.

[3] We do not consider extensions of $\mathcal{N}$ in which only one of the two projections, respectively one of the two coprojections, is present.

oarieli@mta.ac.il

We refer to the language $\mathcal{N}(\backslash, di, ^{-1})$ as the *relation algebra*. For each set F of nonbasic features considered above not containing transitive closure, all path queries expressible in $\mathcal{N}(F)$ are also expressible in the relation algebra [17].

For the purpose of showing the main result, we also consider *conditionals* as nonbasic atomic features in this paper. At the syntactic level, a conditional is an expression denoted by some constant, say c. The semantics of c is given by some (implicit) mapping that associates to each directed graph G a set $c(G)$ of identical pairs of G. Hence, $c(G) \subseteq id(G)$. Informally, $(v, v) \in c(G)$ means that node v "satisfies" c in G. In this paper, we shall use conditionals to eliminate projection subexpression temporarily, as explained in Section 5 and illustrated in Example 2.

Language expressiveness can be considered at the level of path queries and at the level of boolean queries.

Definition 1. *A* path query *q is expressible in a language $\mathcal{N}(F)$ if there exists an expression $e \in \mathcal{N}(F)$ such that, for every graph G, we have $e(G) = q(G)$. Similarly, a* boolean query *q is expressible in $\mathcal{N}(F)$ if there exists an expression $e \in \mathcal{N}(F)$ such that, for every graph G, we have that $e(G)$ is nonempty if and only if $q(G)$ is true. In both cases, we say that q* is expressed by *e.*

In this paper, we are mainly interested in boolean queries. Compared to path queries, this means that we are not interested in the precise set of pairs returned by an expression on a given input graph, but rather in whether or not this set is empty. Hence, if we can establish that adding transitive closure to a language does not increase its expressive power at the level of path queries, this must necessarily also be the case at the level of boolean queries. The converse, however, need not be true. Therefore, studying expressiveness issues is considerably more difficult at the level of boolean queries than at the level of path queries.

To conclude these preliminaries, we formally define what we mean by a *subexpression* of a given expression.

Definition 2. *Let F be a set of nonbasic features, and let e be an expression in $\mathcal{N}(F)$. The set of all* subexpressions *of e, denoted $Sub(e)$, is defined recursively, as follows:*

1. *if e is either R, $\emptyset$, id, di, or a conditional, then $Sub(e) = \{e\}$;*
2. *if "$\diamond$"is either composition or a set operation, and if, for some expressions e_1 and e_2 in $\mathcal{N}(F)$, $e = e_1 \diamond e_2$, then $Sub(e) = Sub(e_1) \cup Sub(e_2) \cup \{e\}$; and*
3. *if "θ" is either projection, coprojection, converse, or transitive closure, and if, for some expression f in $\mathcal{N}(F)$, $e = \theta(f)$, then $Sub(e) = Sub(f) \cup \{e\}$.*

An *atomic subexpression* of an expression is a subexpression that is either "R", "id", "di", or a conditional. For an expression e in the relation algebra with or without transitive closure, we denote by $|e|$ the number of its atomic subexpressions and by $|e|_R$ the number of occurrences of "R" in e.

oarieli@mta.ac.il

3 Relation Algebra Fragments with Coprojection

In this Section, we show that adding transitive closure to a relation algebra fragment in which coprojection can be expressed yields additional expressive power at the level of boolean queries, thus settling the previously open cases for (1) the basic language to which coprojection is added and (2) the basic language to which both coprojection and the diversity relation are added.

A node v of a graph G is a *sink node* if v has no outgoing edges, i.e., if $(v, v) \in \overline{\pi}_1(R)(G)$. We now present an expressibility and an inexpressibility result for a particular graph property stated in terms of sink nodes.

Proposition 1. *The boolean query "There is a non-sink node from which no sink node can be reached" is expressible in* $\mathcal{N}(\overline{\pi}, ^+)$.

Proof. This query returns *true* if and only if $\overline{\pi}_1((R^+ \circ \overline{\pi}_1(R)) \cup \overline{\pi}_1(R)) \neq \emptyset$. □

Using an Ehrenfeucht-Fraïssé argument [16], we can show the following, however (proof omitted).

Proposition 2. *The boolean query "There is a non-sink node from which no sink node can be reached" is not expressible in FO.*

From Propositions 1 and 2 we can now conclude the following.

Theorem 1. *Let F be a set of nonbasic features not containing transitive closure such that coprojection can be expressed in $\mathcal{N}(F)$. Then $\mathcal{N}(F, ^+)$ does* not *collapse to $\mathcal{N}(F)$ at the level of boolean queries.*

Proof. In Propositions 1 and 2, we identified a property of graphs that is expressible in $\mathcal{N}(F, ^+)$ but not in FO. It is well known [24] that the full relation algebra is equivalent to FO^3, both at the level of path queries and at the level of boolean queries. Since FO^3 is a fragment of FO, it follows that the aforementioned property is *not* expressible in $\mathcal{N}(F)$. □

As an immediate corollary, two open cases are now settled.

Corollary 1. *At the level of boolean queries, $\mathcal{N}(\overline{\pi}, ^+)$ does not collapse to $\mathcal{N}(\overline{\pi})$ and $\mathcal{N}(\overline{\pi}, di, ^+)$ does not collapse to $\mathcal{N}(\overline{\pi}, di)$.*

4 Relation Algebra Fragments with at Most Projection and Diversity

This Section is devoted to demonstrating that $\mathcal{N}(\pi, di, ^+)$ collapses to $\mathcal{N}(\pi, di)$ at the level of boolean queries. We start with an introductory example.

Example 1. Consider the expression $e := \pi_1(R^3) \circ R^+ \circ di \circ \pi_2(R) \circ R^2$ in $\mathcal{N}(\pi, di, ^+)$. Let G be a graph. For $e(G)$ to be nonempty, the subexpressions to the right of "di" must return nonempty. Hence, there must exist a chain

oarieli@mta.ac.il

$w_0 \to w_1 \to w_2 \to w_3$ in G. Unless, for each such chain, $w_1 = w_2 = w_3$, it is readily seen that this condition is also sufficient for $e(G) \neq \emptyset$. In the other case, there must exist an edge $v_0 \to v_1$ with a self-loop in v_1 for which $v_1 \neq w_1$ in order for $e(G)$ to be nonempty. It can now be readily verified that, in both cases, $e'(G) \neq \emptyset$, with $e' := \pi_1(R^3) \circ (R \cup R^2) \circ di \circ \pi_2(R) \circ R^2$ in $\mathcal{N}(\pi, di)$. As always $e'(G) \subseteq e(G)$, the converse implication also holds, so $e' \in \mathcal{N}(\pi, di)$ is equivalent to $e \in \mathcal{N}(\pi, di, ^+)$ at the level of boolean queries. □

The argument used to show that transitive closure can be eliminated from the expression in Example 1 is very ad-hoc. Moreover, the considered expression is very simple. We therefore need a general technique to show that $\mathcal{N}(\pi, di, ^+)$ collapses to $\mathcal{N}(\pi, di)$ at the level of boolean queries. In this section, we outline this technique, and, in subsequent sections, we work it out in further detail.

The technique we are about to describe actually works more generally for showing that $\mathcal{N}(F, ^+)$ collapses to $\mathcal{N}(F)$ for all sets of nonbasic features F for which $F \subseteq \{\pi, di\}$. It consists of two steps. Given an expression e in $\mathcal{N}(F, ^+)$,

1. find an expression suff_e in $\mathcal{N}(F)$ such that, for every graph G, $\mathit{suff}_e(G) \neq \emptyset$ implies $e(G) \neq \emptyset$; and
2. find an expression e' in $\mathcal{N}(F)$ that is equivalent to e at the level of boolean queries on all graphs G for which $\mathit{suff}_e(G) = \emptyset$.

It then follows immediately that, on all graphs, e is equivalent to $\mathit{suff}_e \cup e'$ at the level of boolean queries, i.e., for every graph G, $\mathit{suff}_e \cup e'(G) \neq \emptyset$ if and only if $e(G) \neq \emptyset$. Intuitively, $\mathit{suff}_e(G) \neq \emptyset$ is a sufficient condition for $e(G)$ to be nonempty. It therefore suffices to show the collapse on graphs that do not satisfy this condition, i.e., for which $\mathit{suff}_e(G) = \emptyset$. If suff_e is well-chosen, then the latter condition will turn out to be sufficiently restrictive for our purposes.

The first step of the proof procedure is secured by the following result.

Theorem 2. *Let $F \subseteq \{\pi, di\}$. Let e be an expression in $\mathcal{N}(F, ^+)$. Let $\mathit{suff}_{F,e}$ in $\mathcal{N}(F)$ be as tabulated in Table 1, where $m_e = \max(1, |e|_R)$. Then, for every graph G, $\mathit{suff}_{F,e}(G) \neq \emptyset$ implies $e(G) \neq \emptyset$.*

Table 1. Expressions $\mathit{suff}_{F,e}$ in $\mathcal{N}(F)$ for which $\mathit{suff}_{F,e}(G) \neq \emptyset$ implies $e(G) \neq \emptyset$, $F \subseteq \{\pi, di\}$. In these expressions, $m_e = \max(1, |e|_R)$.

F	$\mathit{suff}_{F,e}$
$\emptyset$	R^{m_e}
$\{\pi\}$	R^{m_e}
$\{di\}$	$R^{m_e} \circ di \circ R^{m_e}$
$\{\pi, di\}$	$\pi_1(R^{m_e}) \circ \pi_2(R^{m_e}) \circ di \circ \pi_1(R^{m_e}) \circ \pi_2(R^{m_e})$

Proof. Here, we only sketch the proof for $F = \{\pi, di\}$. So, let e be an expression in $\mathcal{N}(\pi, di, ^+)$ and let G be a graph. We first observe that the condition $\pi_1(R^{m_e}) \circ \pi_2(R^{m_e}) \circ di \circ \pi_1(R^{m_e}) \circ \pi_2(R^{m_e})(G) \neq \emptyset$ is equivalent to the existence of two

oarieli@mta.ac.il

sequences of not necessarily all different nodes $v_{-m_e}, \ldots, v_{-1}, v_0, v_1, \ldots, v_{m_e}$ and $w_{-m_e}, \ldots, w_{-1}, w_0, w_1, \ldots, w_{m_e}$ in G such that, (1) for $i = -m_e, \ldots, m_e - 1$, $(v_i, v_{i+1}) \in R$ and $(w_i, w_{i+1}) \in R$ and (2) $v_0 \neq w_0$. By an inductive argument, we show that, for any union-free expression f in $\mathcal{N}(\pi, di)$ with $|f|_R \leq |e|_R$, f is nonempty on the subgraph of G consisting of the nodes and edges singled out above, as a consequence of which $f(G) \neq \emptyset$. Finally, from e, we construct a union-free expression e' in $\mathcal{N}(\pi, di)$ for which $|e'|_R \leq |e|_R$ and $e'(G) \subseteq e(G)$. It follows now that $e(G) \neq \emptyset$, as had to be shown. □

For the second step, we first of all observe that, for any graph G, and for any nonzero natural number m, $R^m(G) = \emptyset$ implies that $R^m \circ di \circ R^m(G) = \emptyset$, and that $R^m \circ di \circ R^m(G) = \emptyset$ implies that $\pi_1(R^m) \circ \pi_2(R^m) \circ di \circ \pi_1(R^m) \circ \pi_2(R^m) = \emptyset$. Any necessary condition on the graph G for the last expression to return the empty set on G is therefore also a necessary condition for the two other expressions to return the empty set on G.

For our purpose, we extend the notion of directed acyclic graph (DAG).

Definition 3. *An* extended directed acyclic graph *(EDAG) is a (not necessarily connected) DAG to which self-loops may be added provided each path in the DAG contains at most one node with a self-loop. The DAG obtained from an EDAG by removing all self-loops (but not the nodes in which these self-loops occur) is called the* underlying *DAG. The* depth *of an EDAG is the depth of the underlying DAG, i.e., the maximal length of a path in that DAG.*

We now have the following.

Lemma 1. *Let m be a nonzero natural number, and let G be a graph such that $\pi_1(R^m) \circ \pi_2(R^m) \circ di \circ \pi_1(R^m) \circ \pi_2(R^m)(G) = \emptyset$. Then G is an EDAG of depth at most $2m$.*

Proof. If $\pi_1(R^m) \circ \pi_2(R^m) \circ di \circ \pi_1(R^m) \circ \pi_2(R^m)(G) = \emptyset$, then it is the case that, for any two sequences of nodes $v_{-m}, \ldots, v_{-1}, v_0, v_1, \ldots, v_m$ and $w_{-m}, \ldots, w_{-1}, w_0, w_1, \ldots, w_m$ in G such that, for $i = -m, \ldots, m-1$, $(v_i, v_{i+1}) \in R$ and $(w_i, w_{i+1}) \in R$, we have that $v_0 = w_0$ (cf. the proof of Theorem 2). Clearly, this is not the case if G contains either one loop of length at least two; or two self-loops; or a non-selfintersecting path of length at least $2m + 1$. Hence, G is an EDAG of depth at most $2m$. □

Notice that G being an EDAG of depth at most $2m$ is not a sufficient condition for the expression in Lemma 1 to evaluate to the empty set. For instance, an EDAG may contain more than one self-loop in total (at most one on each path in the underlying DAG). Also, a DAG (which is a special case of an EDAG) of depth $2m$ may contain two paths of length $2m$ of which the middle nodes do not coincide. Hence, G being an EDAG of depth at most $2m$ is only a necessary condition for $\pi_1(R^m) \circ \pi_2(R^m) \circ di \circ \pi_1(R^m) \circ \pi_2(R^m)(G) = \emptyset$. For our purposes, however, this is all we need.

Using our earlier observation, we can bootstrap Lemma 1 as follows.

oarieli@mta.ac.il

Proposition 3. *Let $F \subseteq \{\pi, di\}$, and let e be an expression in $\mathcal{N}(F, ^+)$. Let G be a graph such that $suff_{F,e}(G) = \emptyset$. Then G is an EDAG of depth at most $2m_e$.*

Assume that we are given an expression e in $\mathcal{N}(\pi, di, ^+)$ and an EDAG G of depth at most m, with m some nonzero natural number. The remainder of this paper is concerned with proving that there exists a nonzero natural number s depending only on m and e such that $e(G) = \emptyset$ if and only $e'(G) = \emptyset$, where e' is obtained from e by exhaustively replacing each subexpression of the form f^+ by $\bigcup_{i=1}^{s} f^i$.

To achieve this, we intend to show (Proposition 12) that there exists a nonzero natural number s such that, for any node v of G, there exists a subgraph G_v of G containing v which has at most s nodes and satisfies the following property: there exists a node w for which $(v, w) \in e(G)$ if and only if there exists a node w' in G_v for which $(v, w') \in e(G_v)$. To see that this is sufficient for our purposes, assume first that $e(G) = \emptyset$. Then $e'(G) = \emptyset$, since $e'(G) \subseteq e(G)$. Therefore, assume next that $e(G) \neq \emptyset$. Then, for some nodes v and w of G, $(v, w) \in e(G)$. Hence, there exists a node w' in G_v such that $(v, w') \in e(G_v)$. Since G_v has at most s nodes, $e(G_v) = e'(G_v)$. It follows that $e'(G_v) \neq \emptyset$. Since $e'(G_v) \subseteq e'(G)$, it also follows that $e'(G) \neq \emptyset$.

In the remaining sections, we shall establish that such subgraphs G_v exist.

Finally, notice that, whenever e is in $\mathcal{N}(F, ^+)$ with $F \subseteq \{\pi, di\}$, then e' is in $\mathcal{N}(F)$. Hence, our efforts in the context of $\mathcal{N}(\pi, di, ^+)$ also serve to show that any of the languages $\mathcal{N}(F, ^+)$ with $F \subseteq \{\pi, di\}$ collapses to $\mathcal{N}(F)$ at the level of boolean queries.

5 Expressions with Conditionals

To facilitate achieving the goals set at the end of the previous section, we shall first simplify the expressions under consideration. In Section 2, we introduced conditionals, which are constants at the syntactical level, representing at the semantic level functions that associate to each graph a set of identical pairs of that graph. Now, notice that any subexpression of the form $\pi_1(f)$ or $\pi_2(f)$ of an expression in $\mathcal{N}(\pi, di, ^+)$ can be interpreted as a function defining the semantics of some conditional. Given an expression in $\mathcal{N}(\pi, di, ^+)$, we shall therefore as a first step replace all projection subexpressions which themselves do not occur within a projection subexpression by a conditional with the same semantics. In this way, projection is formally eliminated, which simplifies the further development considerably. Once we have a partial result for this case, we will reintroduce the projections and bootstrap the initial result to the desired result.

Example 2. Consider the expression $(R \circ \pi_1((R^3 \circ di \circ \pi_2(R^2) \circ R)^+))^+ \circ R^2$. If we associate a conditional c to $\pi_1((R^3 \circ di \circ \pi_2(R^2) \circ R)^+)$, the expression can be rewritten as $(R \circ c_1)^+ \circ R^2$, i.e., the projection has formally been eliminated.

Therefore, we introduce a finite set of conditionals $\Gamma = \{c_1, \ldots, c_p\}$, and consider the language $\mathcal{N}(\Gamma, di, ^+)$, as well as some of its sublanguages. Later on, we

oarieli@mta.ac.il

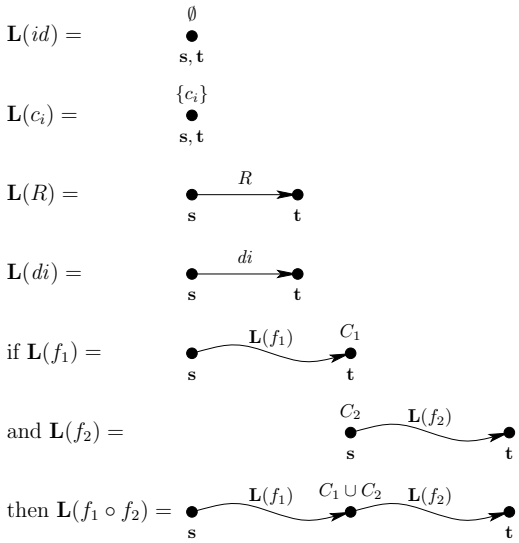


Fig. 1. Definition of the line pattern $\mathbf{L}(f)$ of a union-free expression in $\mathcal{N}(\Gamma, di)$

will choose p as a function of the number of projection subexpressions in the expression under consideration.

A useful property is that, for union-free expressions in $\mathcal{N}(\Gamma, di)$, the presence of a particular pair of nodes of a graph in the output of the expression applied to the graph can be rephrased as the existence of a particular homomorphism from a chain-like directed graph, representing the expression, into the graph.

More concretely, let f be a union-free expression in $\mathcal{N}(\Gamma, di)$. We shall associate a *line pattern* $\mathbf{L}(f)$ with f. This line pattern is a chain-like directed graph in which each edge is labeled with either "R" or "di" and each node is labeled by a (possibly empty) set of conditionals. In addition, each line pattern has one *source node*, labeled $\mathbf{s}$, and one *target node*, labeled $\mathbf{t}$, which may coincide. The precise, inductive, definition is given in Figure 1.

Line patterns are special cases of *graph patterns*. A graph pattern is a directed graph in which each edge is labeled with either "R" or "di" and each node is labeled by a (possibly empty) set of conditionals. At least one node is marked as source, and at least one node is marked as target.

Let $\mathbf{P}$ be a graph pattern, and let G be a directed graph. A mapping h from the nodes of $\mathbf{P}$ to the nodes of G is called a *homomorphism* from $\mathbf{P}$ to G if

1. for each node $\mathbf{v}$ of $\mathbf{P}$, all the conditionals by which $\mathbf{v}$ is labeled are satisfied by $h(\mathbf{v})$ in G;
2. for each edge $(\mathbf{v}, \mathbf{w})$ of $\mathbf{P}$ labeled by "R", $(h(\mathbf{v}), h(\mathbf{w}))$ is an edge of G; and
3. for each edge $(\mathbf{v}, \mathbf{w})$ of $\mathbf{P}$ labeled by "di", $h(\mathbf{v}) \neq h(\mathbf{w})$.

Notice that we use boldface characters for the nodes of line and graph patterns to distinguish them clearly from the nodes of the input graph.

oarieli@mta.ac.il

General graph patterns will be put to use in Section 6 to construct, given an expression e in $\mathcal{N}(\Gamma, di, ^+)$, a natural number m, an EDAG G of depth at most m, and a node v of G, a sequence of subgraphs of G. The number of nodes of these subgraphs can be bounded by a natural number depending only on m and e. This last property is essential for our proof strategy to work, as explained at the end of Section 4.

Turning back to line patterns for now, the following result is obvious.

Proposition 4. *Let G be a directed graph and let f be a union-free expression in $\mathcal{N}(\Gamma, di)$. Let $\boldsymbol{L}(f)$ be the line pattern of f. Let v and w be nodes of G. Then $(v, w) \in f(G)$ if and only if there exists a homomorphism h from $\boldsymbol{L}(f)$ to G with $h(\boldsymbol{s}) = v$ and $h(\boldsymbol{t}) = w$.*

In order to put line patterns to use, we must link expressions in $\mathcal{N}(\Gamma, di, ^+)$ to union-free expressions in $\mathcal{N}(\Gamma, di)$. Thereto, we introduce *trace expressions.*

Definition 4. *Let e be an expression in $\mathcal{N}(\Gamma, di, ^+)$. Then, $\mathcal{T}(e)$, the set of* trace expressions *of e, is defined recursively, as follows:*

- *if e is an atomic expression, then $\mathcal{T}(e) = \{e\}$;*
- $\mathcal{T}(e_1 \cup e_2) = \mathcal{T}(e_1) \cup \mathcal{T}(e_2)$;
- $\mathcal{T}(e_1 \circ e_2) = \{\tau_1 \circ \tau_2 \mid \tau_1 \in \mathcal{T}(e_1) \ \& \ \tau_2 \in \mathcal{T}(e_2)\}$; *and*
- $\mathcal{T}(e^+) = \bigcup_{n>0}\{\tau_1 \circ \cdots \circ \tau_n \mid \forall i = 1, \ldots, n : \tau_i \in \mathcal{T}(e)\}$.

Notice that trace expressions do not contain "$\cup$" and "+".

By a straightforward structural induction, one can show the following.

Proposition 5. *Let e be an expression in $\mathcal{N}(\Gamma, di, ^+)$. Let G be a graph and v and w nodes of G. Then, $(v, w) \in e(G)$ if and only if there exists a trace expression $f \in \mathcal{T}(e)$ such that $(v, w) \in f(G)$.*

The problem with trace expressions is that they may contain a lot of redundancy. Therefore, we define the following notions.

Definition 5. *Let n be a nonzero natural number. An expression g in $\mathcal{N}(\Gamma)$ is n-normal if (1) g is union-free, (2) $|g|_R \leq n$, and (3) a subexpression of g consisting only of "id" conditionals, and composition has at most one occurrence of "id" and at most one occurrence of every conditional.*

Observe that, for all n, "*id*" is always n-normal. We denote the n-normal expressions of $\mathcal{N}(\Gamma)$ by $\mathcal{N}_n^{\text{norm}}(\Gamma)$.

Definition 6. *Let n be a nonzero natural number. An expression f in $\mathcal{N}(\Gamma, di)$ is n-normal if it is of the form $g_1 \circ di \circ g_2 \circ di \circ \cdots \circ g_{k-1} \circ di \circ g_k$, with $g_1, \ldots, g_k \in \mathcal{N}_n^{\text{norm}}(\Gamma)$.*

In particular, all n-normal expressions of $\mathcal{N}(\Gamma)$ are also n-normal expressions of $\mathcal{N}(\Gamma, di)$. We denote the n-normal expressions of $\mathcal{N}(\Gamma, di)$ by $\mathcal{N}_n^{\text{norm}}(\Gamma, di)$. The interest of normal expressions lays in the following proposition.

oarieli@mta.ac.il

Proposition 6. *Let $\Gamma = \{c_1, \ldots, c_p\}$ be a set of conditionals, and let n be a nonzero natural number. Then,*

1. *the number of atomic subexpressions of an expression of $\mathcal{N}_n^{\mathrm{norm}}(\Gamma)$ can be bounded by a number depending only on n and p; and*
2. *the number of expressions in $\mathcal{N}_n^{\mathrm{norm}}(\Gamma)$ is finite, and can be bounded by a number depending only on n and p.*

Given a nonzero natural number n, we now define the set $\mathcal{T}_n^{\mathrm{norm}}(e)$ of *n-normal trace expressions* as the set of all expressions $\mathcal{N}_n^{\mathrm{norm}}(\Gamma, di)$ for which there exists an equivalent expression in $\mathcal{T}(e)$ at the level of path queries.

The following result states that trace expressions can be normalized provided the input graph is an EDAG of bounded depth.

Proposition 7. *Let m be a nonzero natural number, and let e be an expression in $\mathcal{N}(\Gamma, di, ^+)$. Then, there exists a nonzero natural number n depending only on m and e such that, for every EDAG G of depth at most m, and for every pair of nodes v and w of G, $(v, w) \in e(G)$ if and only if there exists an n-normal trace expression f in $\mathcal{T}_n^{\mathrm{norm}}(e)$ for which $(v, w) \in f(G)$.*

6 Canonical Subgraphs

Given a set of conditionals $\Gamma = \{c_1, \ldots, c_p\}$, a natural number n, a directed graph G, and a node v of G, we shall now define a sequence of so-called n-canonical subgraphs $G_0^v, G_1^v, G_2^v, \ldots$ of order $0, 1, 2, \ldots$. (In the notation, we shall leave Γ and n implicit.)

Important for our purpose is that it will turn out that the number of nodes of each of these n-canonical subgraphs depends only on its order and on p and n, but *not* on the particular graph G or the particular node v under consideration.

We start by defining G_0^v.

Thereto, let g be an expression in $\mathcal{N}_n^{\mathrm{norm}}(\Gamma)$. We define $\mathfrak{P}(g)$ to be the set of graph patterns that can be obtained from $\mathbf{L}(g)$ in the following way:

1. Start with one, two, three, or four pairwise disjoint copies of $\mathbf{L}(g)$.
2. Optionally, merge some of the source nodes of these copies.
3. Optionally, merge some of the target nodes of these copies.
4. Optionally, connect some of the remaining source nodes by "*di*" edges.
5. Optionally, connect some of the remaining target nodes by "*di*" edges.

Observe that the line pattern $\mathbf{L}(g)$ itself is always in $\mathfrak{P}(g)$.

Figure 2 shows a more representative example of a graph pattern that belongs to $\mathfrak{P}(g)$.

Now, let $\mathbf{P}$ be a graph pattern in $\mathfrak{P}(g)$, and let v be a node of G. With $\mathbf{P}$, we associate a minimal (in number of elements) set $\mathfrak{H}_v(\mathbf{P})$ of homomorphisms from $\mathbf{P}$ to G satisfying the following conditions:

oarieli@mta.ac.il

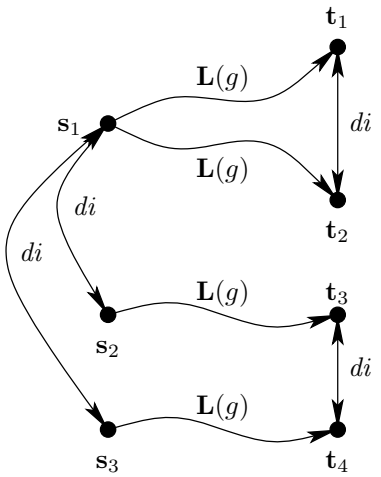


Fig. 2. Example of a graph pattern in $\mathfrak{P}(g)$

1. if there exists a homomorphism from $\mathbf{P}$ to G, then $\mathfrak{H}_v(\mathbf{P}) \neq \emptyset$;
2. if, for an arbitrary node $\mathbf{v}$ of $\mathbf{P}$, there exist two homomorphisms from $\mathbf{P}$ to G mapping $\mathbf{v}$ to different nodes of G, then $\mathfrak{H}_v(\mathbf{P})$ contains two homomorphisms from $\mathbf{P}$ to G mapping $\mathbf{v}$ to different nodes of G;
3. if $\mathbf{P}$ has a single source node $\mathbf{s}$ and there exists a homomorphism from $\mathbf{P}$ to G mapping $\mathbf{s}$ to v, then $\mathfrak{H}_v(\mathbf{P})$ contains such a homomorphism;
4. if $\mathbf{P}$ has a single target node $\mathbf{t}$ and there exists a homomorphism from $\mathbf{P}$ to G mapping $\mathbf{t}$ to v, then $\mathfrak{H}_v(\mathbf{P})$ contains such a homomorphism;

For a good understanding, we first observe the following:

- Given $\mathbf{P}$, G, and v, we *choose* a minimal set of homomorphisms $\mathfrak{H}_v(\mathbf{P})$ satisfying the above conditions. In other words, it is to be expected that, in general, several sets of homomorphisms satisfy the above conditions. From these, we pick one arbitrarily, and denote it by $\mathfrak{H}_v(\mathbf{P})$.
- The definition of $\mathfrak{H}_v(\mathbf{P})$ refers explicitly to v only if $\mathbf{P}$ has either a single source node, or a single target node, or both. In all other cases, we may therefore choose $\mathfrak{H}_v(\mathbf{P})$ independent of v.

We are now ready to define G_0^v, the basic n-canonical subgraph of order 0:

$$G_0^v = \bigcup_{g \in \mathcal{N}_n^{\mathrm{norm}}(\Gamma)} \bigcup_{\mathbf{P} \in \mathfrak{P}(g)} \bigcup_{h \in \mathfrak{H}_v(\mathbf{P})} h(\mathbf{P}).$$

In the above formula, $h(\mathbf{P})$ must be understood as the subgraph of G the set of nodes of which is $\{h(\mathbf{v}) \mid \mathbf{v} \text{ is a node of } \mathbf{P}\}$ and the set of edges of which is $\{(h(\mathbf{v}), h(\mathbf{w})) \mid (\mathbf{v}, \mathbf{w}) \text{ is an } R\text{-labeled edge of } \mathbf{P}\}$. The basic n-canonical subgraph of order 0 is then defined as a union of some of these subgraphs, where

oarieli@mta.ac.il

this union must be interpreted componentwise, i.e., the set of nodes and the set of edges of this union are the union of the sets of nodes and the union of the sets of edges of the subgraphs involved.

We point out at this stage that if a node v of G satisfies a conditional c, and G' is a subgraph of G containing v, then a priori v does not have to satisfy c in G'. We shall therefore avoid evaluating expressions over subgraphs of G (in particular, the canonical ones), until we reinterpret conditionals as the projection subexpressions for which they actually stand, in Section 8.

At this point, several aspects of the definition of the basic n-canonical subgraph of order 0 have been left unexplained, in particular,

- the definition of the set of graph patterns $\mathfrak{P}(g)$ for $g \in \mathcal{N}_n^{\text{norm}}(\Gamma)$, and, more specifically, why up to four copies of the line pattern $\mathbf{L}(g)$ are allowed in such a graph pattern; and
- the definition of the set of homomorphisms $\mathfrak{H}_v(\mathbf{P})$ for $\mathbf{P} \in \mathfrak{P}(g)$.

These definitions are tailored to make the key results Propositions 10 and 11 in Section 7 work, as is explained in that section. The essence is that, given an n-normal trace expression f in $\mathcal{T}_n^{\text{norm}}(e)$ and a homomorphism h from $\mathbf{L}(f)$ to G, we wish to show via an inductive process that there also exists such a homomorphism of which the image is fully contained in one of the n-canonical subgraphs of order 0. On the one hand, we must ensure that the n-canonical subgraphs of order 0 are sufficiently large for this process to work, but, on the other hand, we must also ensure that their size can be bounded by a bound not depending on the size of G (see Proposition 9, below). Obtaining this delicate balance is what led to the definition above.

However, Propositions 10 and 11 are but the first albeit important step in proving the collapse of $\mathcal{N}(\pi, di, ^+)$ to $\mathcal{N}(\pi, di)$. Indeed, the conditionals represent projection conditions, and the operands of these projections may in turn contain projection conditions.

To accommodate this, we next define $G_1^v, G_2^v, \ldots$, the n-canonical subgraphs of G of order $1, 2, \ldots$, with the following inductive rule. For $i > 0$,

$$G_i^v = G_0^v \cup \left(\bigcup_{w \text{ node of } G_0^v} G_{i-1}^w \right).$$

The following property of n-canonical subgraphs is straightforward.

Proposition 8. *Given a set of conditionals $\Gamma = \{c_1, \ldots, c_p\}$, a nonzero natural number n, and a directed graph G, we have, for every node v of G and for $i = 0, 1, 2, \ldots$, that G_i^v is a subgraph of G_{i+1}^v.*

The n-canonical subgraphs of G of higher order are put to use in Section 8, more in particular in Proposition 12.

oarieli@mta.ac.il

For the remainder of the exposition, it is important that we can also provide bounds on the sizes of the n-canonical subgraphs of G.

Proposition 9. *Given a set of conditionals $\Gamma = \{c_1, \ldots, c_p\}$, a nonzero natural number n, and a directed graph G, we have, for every node v of G and for $i = 0, 1, 2, \ldots$, that the number of nodes in G_i^v can be bounded by a number depending only on n, p, and i.*

7 The Key Result

Let $\Gamma = \{c_1, \ldots, c_p\}$ be a set of conditionals. The key results on which the second step in our proof strategy for the collapse of $\mathcal{N}(\Gamma, di, ^+)$ to $\mathcal{N}(\Gamma, di)$ at the boolean level (cf. item 2 on p. 130 and the concluding paragraphs of Section 4) rely, are the following.

Proposition 10. *Let m be a nonzero natural number, and let e be an expression in $\mathcal{N}(\Gamma, di, ^+)$. Then, there exists a nonzero natural number n depending only on m and e such that, for every EDAG G of depth at most m, and for every node v of G, if there exists a node w in G such that $(v, w) \in e(G)$, then there exists an n-normal trace expression f in $\mathcal{T}_n^{\mathrm{norm}}(e)$ and a homomorphism h from $\boldsymbol{L}(f)$ to G such that $h(\boldsymbol{s}) = v$ and $h(\boldsymbol{L}(f))$ is contained in G_0^v, with $\boldsymbol{s}$ the source node of the line pattern $\boldsymbol{L}(f)$ and G_0^v the basic n-canonical subgraph of G.*

Proposition 11. *Let m be a nonzero natural number, and let e be an expression in $\mathcal{N}(\Gamma, di, ^+)$. Then, there exists a nonzero natural number n depending only on m and e such that, for every EDAG G of depth at most m, and for every node w of G, if there exists a node v in G such that $(v, w) \in e(G)$, then there exists an n-normal trace expression f in $\mathcal{T}_n^{\mathrm{norm}}(e)$ and a homomorphism h from $\boldsymbol{L}(f)$ to G such that $h(\boldsymbol{t}) = w$ and $h(\boldsymbol{L}(f))$ is contained in G_0^w, with $\boldsymbol{t}$ the target node of the line pattern $\boldsymbol{L}(f)$ and G_0^w the basic n-canonical subgraph of G.*

It is important to notice here that the homomorphism h in Propositions 10 and 11 need not be a homomorphism from $\mathbf{L}(f)$ to G_0^v, respectively G_0^w. If this were the case, then, by Proposition 4, $(v, w) \in e(G_0^v)$, respectively $(v, w) \in e(G_0^w)$, and we would have found the subgraphs G_v of G we set out to find at the end of Section 4 to achieve the second step of our proof strategy. However, this is in general not the case, the reason being that conditionals are in general not preserved under taking subgraphs. Indeed, if z is a node of G such that $(z, z) \in c_i(G)$, $1 \leq i \leq p$, then it does not follow that, necessarily, $(z, z) \in c_i(G_0^v)$. As mentioned, the case that we are interested in is the case where the conditionals are in fact projection conditions. These have the property of being monotone. To guarantee the above implication, we will therefore have to extend the subgraph G_0^v, and that is where the normal subgraphs of higher order come in play, at a later stage of our development, in Section 8.

Because of the strong analogy between both Propositions, we shall focus here on the proof of Proposition 10. It can be easily seen that Proposition 10 follows from Propositions 7 and 4, provided we can prove the following lemma.

oarieli@mta.ac.il

Lemma 2. *Let G be a directed graph, let n be a nonzero natural number, and let f be an n-normal expression in $\mathcal{N}_n^{\mathrm{norm}}(\Gamma, di)$. Let v be a node of G. If there exists a homomorphism h from $\boldsymbol{L}(f)$ to G such that $h(\boldsymbol{s}) = v$, with $\boldsymbol{s}$ the source node of $\boldsymbol{L}(f)$, then there exists a homomorphism h' from $\boldsymbol{L}(f)$ to G such that $h'(\boldsymbol{s}) = v$ and $h'(\boldsymbol{L}(f))$ is contained in G_0^v, with G_0^v the basic n-canonical subgraph of G.*

If we write $f = g_1 \circ di \circ g_2 \circ di \circ \cdots \circ g_{n-1} \circ di \circ g_n$, with $g_1, \ldots, g_n \in \mathcal{N}_m^{\mathrm{norm}}(\Gamma)$, a sensible way to prove Lemma 2 is to consider the expressions $f_i = g_1 \circ di \circ g_2 \circ di \circ \cdots \circ g_{i-1} \circ di \circ g_i$, for $i = 1, \ldots, n$, and to prove the Lemma by induction on i. The basis of the induction, $i = 1$, is straightforward from the construction of the subgraph G_0^v. Thus suppose that, for $1 < i \leq n$, we have established the existence of a homomorphism h'_{i-1} from $\mathbf{L}(f_{i-1})$ to G such that $h'_{i-1}(\mathbf{s}) = v$ ($\mathbf{s}$ being the source node of $\mathbf{L}(f_{i-1})$) and $h'_{i-1}(\mathbf{L}(f_{i-1}))$ is contained in G_0^v. We would like to extend h'_{i-1} to a homomorphism h'_i from $\mathbf{L}(f_i)$ to G such that $h'_i(\mathbf{L}(f_i))$ is contained in G_0^v. Thus, consider $\mathbf{L}(g_i)$, which is a subpattern of $\mathbf{L}(f_i)$. The restriction of h to the nodes of $\mathbf{L}(g_i)$ is a homomorphism from $\mathbf{L}(g_i)$ to G. Hence, $\mathfrak{H}_v(\mathbf{L}(g_i))$ contains a homomorphism $h_{\mathbf{L}(g_i)}$ from $\mathbf{L}(g_i)$ to G, and, by construction of G_0^v, $h_{\mathbf{L}(g_i)}(\mathbf{L}(g_i))$ is contained in G_0^v. Now, let $\mathbf{t}_{i-1}$ be the target node of $\mathbf{L}(f_{i-1})$ and $\mathbf{s}_i$ the source node of $\mathbf{L}(g_i)$. If $h'_{i-1}(\mathbf{t}_{i-1}) \neq h_{\mathbf{L}(g_i)}(\mathbf{s}_i)$, the extension is straightforward. However, we cannot exclude that $h'_{i-1}(\mathbf{t}_{i-1}) = h_{\mathbf{L}(g_i)}(\mathbf{s}_i)$. If this is the case, it may even be so that $h_{\mathbf{L}(g_i)}$ is the *only* homomorphism mapping $\mathbf{L}(g_i)$ to G. Then, we cannot even consider an alternative homomorphism from $\mathbf{L}(g_i)$ to G to make our extension strategy work.

However, we can avoid this pitfall by proving a slightly stronger statement.

Lemma 3. *Let G be a directed graph, let n be a nonzero natural number, and let f be an n-normal expression in $\mathcal{N}_n^{\mathrm{norm}}(\Gamma, di)$. Let v be a node of G, and let G_0^v be the basic n-canonical subgraph of G. Then,*

1. *if there exist homomorphisms h_1 and h_2 from $\boldsymbol{L}(f)$ to G such that $h_1(\boldsymbol{s}) = h_2(\boldsymbol{s}) = v$ and $h_1(\boldsymbol{t}) \neq h_2(\boldsymbol{t})$, with $\boldsymbol{s}$ and $\boldsymbol{t}$ the source and target nodes of $\boldsymbol{L}(f)$, then there exist homomorphisms h'_1 and h'_2 from $\boldsymbol{L}(f)$ to G such that $h'_1(\boldsymbol{s}) = h'_2(\boldsymbol{s}) = v$, $h'_1(\boldsymbol{t}) \neq h'_2(\boldsymbol{t})$, and $h'_1(\boldsymbol{L}(f))$ and $h'_2(\boldsymbol{L}(f))$ are both contained in G_0^v;*
2. *otherwise, if there exists a homomorphism h from $\boldsymbol{L}(f)$ to G such that $h(\boldsymbol{s}) = v$, with $\boldsymbol{s}$ the source node of $\boldsymbol{L}(f)$, then there exists a homomorphism h' from $\boldsymbol{L}(f)$ to G such that $h'(\boldsymbol{s}) = v$ and $h(\boldsymbol{L}(f))$ is contained in G_0^v.*

The proof goes along the lines of the sketch we gave of the (failed) proof for Lemma 2. In the induction step, we may be in Case 1 or Case 2 of Lemma 2, and to carry out the inductive argument, we may be in Case 1 or Case 2 as far as the induction hypothesis is concerned, giving rise to four possible combinations we need to consider. However, when we are in Case 2 as far as the induction hypothesis is concerned, then, compared to our naive attempt to prove Lemma 2 directly, we can make use of the additional information that *all* homomorphisms from the line pattern under consideration map the target node to the same node of G, for, otherwise, we would be in Case 1. This additional information will prevent us from getting stuck in this case.

oarieli@mta.ac.il

Each time we get a conflict of the sort described in the failed direct proof for Lemma 2, we will create a graph pattern by combining the given information on the existence of homomorphisms from the line segment under consideration to G with the (not directly usable) homomorphisms from this line segment to G of which the image is fully contained in G_0^v. We will reflect our knowledge on the equality or distinctness of nodes in the images of the various homomorphism by merging the corresponding nodes in the graph pattern (in the case of equality) or connecting these nodes by "*di*" edges (in the case of distinctness). This will result in a graph pattern such as the one shown in Figure 2. As, by construction, this graph pattern can be mapped homomorphically to G, it can also be mapped homomorphically to G in such a way that the image is contained in G_0^v, provided the graph pattern does not contain more than four pairwise disjoint copies of the line segment under consideration. It turns out that, in each of the cases we must consider, this is indeed so. The richer information we obtain from the existence of a homomorphism mapping the graph pattern within G_0^v as opposed to the existence of a homomorphism just mapping the line pattern within G_0^v turns out to be sufficient to carry out the inductive step successfully.

8 The Collapse

We are now ready to deal with expressions in $\mathcal{N}(\pi, di, ^+)$ and bootstrap Propositions 10 and 11 by considering that conditionals stand for projection subexpressions. We recall that the homomorphism h in the statements of these propositions is a homomorphism from $\mathbf{L}(f)$ to G such that $h(\mathbf{s}) = v$ and $h(\mathbf{L}(f))$ is contained in G_0^v, but not necessarily a homomorphism from $\mathbf{L}(f)$ to G_0^v, the reason being that a node of G_0^v satisfying a particular conditional within G does not have to satisfy the same conditional within G_0^v. Using that the conditionals stand for projection subexpressions, and using the monotonicity of the projection operator, we are able to establish that G_0^v can be extended to a higher-order canonical subgraph of G, say G_i^v, such that h is also a homomorphism from $\mathbf{L}(f)$ to G_i^v. Only then will we be able to conclude that $(v, h(\mathbf{t})) \in e(G_i^v)$, with $\mathbf{t}$ the target node of $\mathbf{L}(f)$ and can we complete our argument.

For this purpose, we first define the π-*nesting depth* $\mathrm{depth}_\pi(e)$ of an expression e in $\mathcal{N}(\pi, di, ^+)$ as follows, inductively:

- if e is in $\mathcal{N}(di, ^+)$, then $\mathrm{depth}_\pi(e) = 0$;
- $\mathrm{depth}_\pi(\pi_1(e)) = \mathrm{depth}_\pi(\pi_2(e)) = \mathrm{depth}_\pi(e) + 1$;
- $\mathrm{depth}_\pi(e_1 \cup e_2) = \max(\mathrm{depth}_\pi(e_1), \mathrm{depth}_\pi(e_2))$;
- $\mathrm{depth}_\pi(e_1 \circ e_2) = \max(\mathrm{depth}_\pi(e_1), \mathrm{depth}_\pi(e_2))$; and
- $\mathrm{depth}_\pi(e^+) = \mathrm{depth}_\pi(e)$.

With every subexpression $\pi_i(f)$, $i = 1, 2$, of e, we can associate a conditional the semantics of which is precisely described by this subexpression $\pi_i(f)$. We denote the set of all these conditionals by $\Pi(e)$.

We can now show the following.

oarieli@mta.ac.il

Proposition 12. *Let m be a nonzero natural number, and let e be an expression in $\mathcal{N}(\pi, di, ^+)$. Let $\ell := depth_\pi(e)$. Then, there exists a nonzero natural number n depending only on m and e such that, for every EDAG G of depth at most m, and, for every node v of G, if there exists a node w in G such that $(v, w) \in e(G)$, then there exists a node w' in G^v_ℓ such that $(v, w') \in e(G^v_\ell)$, with G^v_ℓ the n-canonical subgraph of G of order ℓ for the set of conditionals $\Gamma := \Pi(e)$.*

Proposition 12 is shown by proving that an extended version of it holds for every subexpression of e, by induction on its π-nesting depth. Propositions 10 and 11 play a key role in this, where the former is needed to deal with the first projection and the latter to deal with the second projection. Notice that, for the expression e itself, Proposition 10 already yields that, for some n-normal trace expression f in $\mathcal{T}^{\text{norm}}_n(e)$, there exists a homomorphism h from the line pattern $\mathbf{L}(f)$ to G such that $h(\mathbf{s}) = v$ and $h(\mathbf{L}(f))$ is contained in G^v_0, with $\mathbf{s}$ the source node of $\mathbf{L}(f)$ and G^v_0 the basic n-canonical subgraph of G. It now turns out that the G^v_ℓ, the n-canonical subgraph of G of order ℓ, is an extension of G^v_0 for which each node of G^v_0 satisfying some conditional of $\Pi(e)$ in G also satisfies this conditional in G^v_ℓ, but not necessarily in G^v_0! Hence, h, while in general not a homomorphism from $\mathbf{L}(f)$ to G^v_0, is a homomorphism from $\mathbf{L}(f)$ to G^v_ℓ, and we can then invoke Propositions 7 and 4 to obtain the conclusion of Proposition 12.

Now, from Proposition 9, it immediately follows that we can bound the number of nodes in G^v_ℓ by a number s depending only on m and e. Hence, we have all the ingredients needed to complete the second step of our proof strategy as explained at the end of Section 4, and we may thus conclude the following.

Theorem 3. *Let m be a nonzero natural number, and let e be an expression in $\mathcal{N}(\pi, di, ^+)$. Then, there exists a nonzero number s depending only on m and e such that, for every EDAG G with depth at most m, $e(G) \neq \emptyset$ if and only if $e'(G) \neq \emptyset$, where e' is the expression in $\mathcal{N}(\pi, di)$ obtained from e by exhaustively replacing each subexpression of the form f^+ by $\bigcup_{i=1}^{s} f^i$.*

Since the parameter s, the bound on the size of the graphs G^v_ℓ in Proposition 12, is of very high complexity in m, it may require very large graphs G before the difference between G and its subgraphs G^v_ℓ becomes significant.[4]

Combining Theorems 2 and 3, we see that $\mathcal{N}(\pi, di, ^+)$ collapses to $\mathcal{N}(\pi, di)$ at the level of boolean queries. Furthermore, if F is a subset of $\{\pi, di\}$ and e is more specifically an expression of $\mathcal{N}(F, ^+)$, then it follows that the expression e' defined in Theorem 3 is more specifically in $\mathcal{N}(F)$. From our proof, we may therefore also conclude the following.

Corollary 2. *Let $F \subseteq \{\pi, di\}$. Then $\mathcal{N}(F, ^+)$ collapses to $\mathcal{N}(F)$ at the level of boolean queries.*

[4] For the same reason, it was not possible to "discover" Proposition 12 and the ensuing Theorem 3 by looking at simple examples.

oarieli@mta.ac.il

9 Conclusions and Future Work

We now have a complete understanding of the impact of adding transitive closure to the relation algebra fragments considered. While it is well-known that transitive closure adds expressive power to all fragments at the level of path queries [4], and the same was established in previous work of the present authors [10] at the level of boolean queries on labeled graphs (multiple input relations), we have now established, in contrast, that, while adding transitive closure adds expressive power to most relation algebra fragments at the level of boolean queries on unlabeled graphs (a single input relation), it does *not* add expressive power to $\mathcal{N}(F)$, with F a set of nonbasic features, if and only if $F \subseteq \{\pi, di\}$.

Towards future work, one may investigate similar problems for other logics. An operation we did not consider, for instance, is residuation. Residuation [22] is similar to the standard relational division operation in databases, and corresponds to the set containment join [18].

References

1. RDF primer (2004), http://www.w3.org/TR/rdf-primer/
2. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann (1999)
3. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley, Reading (1995)
4. Aho, A.V., Ullman, J.D.: The universality of data retrieval languages. In: Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, San Antonio, Texas, pp. 110–120 (January 1979)
5. Angles, R., Gutiérrez, C.: Survey of graph database models. ACM Comput. Surv. 40(1), 1–39 (2008)
6. Baader, F., Calvanese, D., McGuiness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
7. Benedikt, M., Fan, W., Kuper, G.M.: Structural Properties of XPath Fragments. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 79–95. Springer, Heidelberg (2002)
8. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semantic Web Inf. Syst. 5(3), 1–22 (2009)
9. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press (2001)
10. Fletcher, G.H.L., Gyssens, M., Leinders, D., Van den Bussche, J., Van Gucht, D., Vansummeren, S., Wu, Y.: Relative expressive power of navigational querying on graphs. In: Milo, T. (ed.) ICDT, pp. 197–207. ACM (2011)
11. Florescu, D., Levy, A., Mendelzon, A.: Database techniques for the World-Wide Web: A survey. SIGMOD Record 27(3), 59–74 (1998)
12. Franklin, M.J., Halevy, A.Y., Maier, D.: From databases to dataspaces: a new abstraction for information management. SIGMOD Record 34(4), 27–33 (2005)
13. Gyssens, M., Paredaens, J., Van Gucht, D., Fletcher, G.H.L.: Structural characterizations of the semantics of XPath as navigation tool on a document. In: Vansummeren, S. (ed.) PODS, pp. 318–327. ACM (2006)
14. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)

oarieli@mta.ac.il

15. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space, 1st edn. Synthesis Lectures on the Semantic Web: Theory and Technology, vol. 1. Morgan & Claypool Publishers (February 2011)
16. Libkin, L.: Elements of Finite Model Theory. Springer, Berlin (2004)
17. Maddux, R.D.: Relation Algebras. Elsevier, Amsterdam (2006)
18. Mamoulis, N.: Efficient processing of joins on set-valued attributes. In: Proceedings ACM SIGMOD International Conference on Management of Data, pp. 157–168 (2003)
19. Marx, M., Venema, Y.: Multi-Dimensional Modal Logic. Springer, Heidelberg (1997)
20. Marx, M.: Conditional XPath. ACM Trans. Database Syst. 30(4), 929–959 (2005)
21. Marx, M., de Rijke, M.: Semantic characterizations of navigational XPath. SIGMOD Record 34(2), 41–46 (2005)
22. Pratt, V.R.: Origins of the calculus of binary relations. In: Proceedings 7th Annual IEEE Symposium on Logic in Computer Science, pp. 248–254 (1992)
23. Tarski, A.: On the calculus of relations. J. of Symbolic Logic 6(3), 73–89 (1941)
24. Tarski, A., Givant, S.: A Formalization of Set Theory without Variables. American Mathematical Society (1987)
25. Wu, Y., Van Gucht, D., Gyssens, M., Paredaens, J.: A study of a positive fragment of Path queries: Expressiveness, normal form and minimization. Comput. J. 54(7), 1091–1118 (2011)

oarieli@mta.ac.il

Using Functional Dependencies for Reducing the Size of a Data Cube

Eve Garnaud[1], Sofian Maabout[1,2], and Mohamed Mosbah[1]

[1] University of Bordeaux, LaBRI, CNRS, UMR 5800, F-33400 Talence. France
{garnaud,maabout,mosbah}@labri.fr
[2] INRIA- Bordeaux Sud Ouest

Abstract. Functional dependencies (FD's) are a powerful concept in data organization. They have been proven very useful in *e.g.,* relational databases for reducing data redundancy. Little work however has been done so far for using them in the context of data cubes. In the present paper, we propose to characterize the parts of a data cube to be materialized with the help of the FD's present in the underlying data. For this purpose, we consider two applications: (i) how to choose the *best cuboids* of a data cube to materialize in order to guarantee a fixed performance of queries and, (ii) how to choose the *best tuples*, hence partial cuboids, in order to reduce the size of the data cube without loosing information. In both cases we show how FD's are fundamental.

1 Introduction

Since the introduction of the data cube operator in [17], it has been recognized that in real life applications, it was unfeasible to store its whole content. Thus, many proposals have been made in order to find the *most* interesting part of it to be materialized. We distinguish two major directions that have been pursued for this purpose. The first one, we qualify it as physical design oriented, tries to select the parts that reduce the cost of the queries under some budget constraint like memory space and/or allowed time for materialized data maintenance, *e.g.,* [20,31,2,24]. The second orientation view the data cube as a set of facts (tuples) and tries to remove those that can be deduced using some reasoning mechanism. This is often qualified as semantic summarization, *e.g.,* [33,23,9]. The major contributions of previous works have been in algorithmic techniques by providing exact, approximate or even heuristic solutions.

On the other hand, we note that since the relational model has been introduced [14], functional dependencies (FD's) have played a fundamental role in *normalization theory* so as to characterize the logical organization of redundancy free data bases [1,25]. So it seems natural to investigate how FD's can be used in the data cubes context. This is the purpose of our present work. More precisely, we make the following contributions:

– We show how to characterize the views to be selected for materialization with exact or approximative FD's while guaranteeing the performance of queries evaluation.

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 144–163, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

– By extending the class of queries we want to optimize, we show how *conditional* FD's [7] can be used to characterize the minimal set of tuples to be materialized, and
– we show how FD's can be used to characterize semantic summaries by considering two well known techniques: condensed cubes [33] and quotient cubes [23].

Paper organization. Section 2 reviews some related work and Section 3 introduces the necessary definitions and notations as well as the general problem we want to solve. In 4, we consider a class of queries and show how to select the cuboids to be materialized with the help of exact and approximate FD's. The class of queries we consider is then extended in 5 and there we characterize the tuples from different cuboids that should are be materialized and this time with the help of conditional FD's. Finally, we show in Section 6 how conditional FD's can be used in order to express the condensed and quotient cubes concepts in 7. We terminate by a conclusion and giving indications of future work.

2 Related Work

Materialized views have been recognized as an effective way to optimize queries since a long time, *e.g.,* see [5]. All modern data base management systems are equipped with tools helping the data base administrator to tune their applications and providing suggestions such as the creation of indexes and/or materialized views, *e.g.,* [29,26]. A comprehensive survey of the algorithmic techniques these softwares implement can be found in [8]. All these techniques consider essentially the same problem: which views to materialize in order to minimize queries evaluation while respecting a storage space budget. Several propositions have been made to solve that problem. [24] proposed to use Integer Linear Programming techniques to find an exact solution with the help of modern solver softwares. Unfortunately, since the problem is NP-Hard, this method is not viable when the number of dimensions is even moderate[1]. Hence, approximate or even heuristic solutions have been proposed, *e.g.,* [20] used a greedy algorithm guaranteeing a constant *gain* approximation factor. More precisely, let $w1, w2$ be the queries evaluation costs when respectively no view is materialized and when some views are materialized. Clearly $w2 \leq w1$ and the gain is, by definition equal to $w1 - w2$. Actually, $w1$ can be seen as the worst cost. However, as it has been shown later in [22], an approximation constant of the gain does not provide any guarantee w.r.t. the queries evaluation cost, *i.e.,* the query cost can be arbitrary larger than that we would obtain with the exact solution. [31] proposed an heuristic solution consisting simply in picking the cuboids w.r.t. their ascending sizes until the available storage space is reached. The authors show

[1] Recall that the problem is exponential in the number of dimensions. Our previous experiments using Cplex solver show that with 10 dimensions, it takes more than one hour to get a solution.

oarieli@mta.ac.il

experimentally that their solution is competitive comparatively to [20] while it has a lower complexity. One should note that this method does not provide any guarantee about the cost of queries whereas it is our main objective. All the above mentioned proposals suffer from a common limitation due to the way they formalize th optimization problem. Indeed, all of them consider a set of *target* queries to be optimized (a workload $\mathcal{Q}$) and they try to minimize the *sum* of the queries costs. The problem with this formalization is that minimizing the total cost of a set of queries does not imply that individual queries will be optimized. Very few solutions, *e.g.,* [11], guarantee that each query will eventually be optimized. Based on this observation, [18,19] considered query evaluation performance as being the constraint to be satisfied and the objective, just like ours, is to minimize the storage space of the solution. They provide algorithmic techniques for computing solutions with proven approximation factors. Even if our objective is the same, the present paper is rather focused in the usage of FD's for *characterizing* the solution instead of how to compute it. Our hope is that this will shed light on the central role played by FD's thus one could reuse well established results about them in the context of data cubes. It is worthwhile noting the work presented in [27] where the authors proposed also to use functional dependencies in order to reduce the *sparsity* of cubes. One main difference of that work with ours is that we consider a ROLAP implementation, *i.e.,* each cuboid is materialized as a relational table, while [27] considers a MOLAP implementation *i.e.,* the whole cube is stored as a multidimensional array. Hence, they rather deal with the reduction of empty cells in those arrays. Another difference is that we consider not only exact dependencies but also approximate ones.

In the summarization context, [23] proposed the quotient cube structure. They create classes of equivalent tuples w.r.t. their measures. Intuitively two tuples of the data cube are equivalent if they have the same measure and one can drill from one tuple to the other. Thus, it is needless to store both of them. To be more precise, each equivalence class is represented by two sets of tuples: the upper and the lower bounds of the class. Each tuple belonging between them, its value can be recovered without any computation and thus is not stored. A more detailed description of quotient cubes is provided in Section 7. A quite similar approach has been proposed independently in [9]. Condensed cubes [33] took a different approach in that the obtained summary does not depend on the computed measures but on the relationships between dimensions values. Again, this structure is described in detail in Section 6. We show how both of these proposals can be expressed using *conditional* FD's.

Finally, FD's and their variants have been recognized as a strong tool for data base normalization since the relational model has been proposed. While FD's have been used in vertical decomposition [13,4], conditional FD's (CFD's) helped in defining horizontal partitioning [6]. Recently, CFD's have gained a new revival and attracted a lot of research because of their ability to detect and help to correct dirty data [7,15,12].

oarieli@mta.ac.il

3 Preliminaries

Let $T(Dim_1, \ldots, Dim_n, M)$ be a fact table where attributes $Dim_i, 1 \leq i \leq n$ are *dimensions* and M is a *measure*. The cube $Cube$ obtained from T is defined by the query

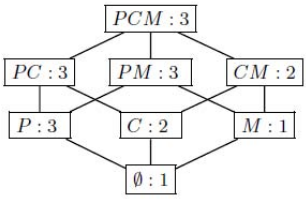
```
SELECT    Dim_1, ..., Dim_n, agg(M)
FROM      T
GROUP BY CUBE Dim_1, ..., Dim_n
```

$agg(M)$ is an algebraic aggregate function, *e.g.,* `COUNT`, `MIN`, `MAX` or `SUM`. The above query is equivalent to the union of the 2^n group-by queries with each subset of $\{Dim_1, \ldots, Dim_n\}$ each of which defines a *cuboid*. $Dim(Cube)$ denotes the set $\{Dim_1, \ldots, Dim_n\}$. If c is a cuboid of $Cube$ then $Dim(c)$ denotes the dimensions of c. Following the OLAP spirit, we consider that users may interact with the cube in an unpredictable way so we do not consider a workload of queries. Let c and c' be two cuboids of $Cube$. We note $c \preceq c'$ iff $Dim(c) \subseteq Dim(c')$. $< Cube, \preceq >$ is the lattice we will consider. For a cuboid c of C we define: (i) its set of ancestors $A_c = \{c' \in Cube |\ c \preceq c'\}$, (ii) its set of descendants: $D_c = \{c' \in Cube |\ c' \preceq c\}$ and (iii) its set of parents: $P_c = \{c' \in A_c\ st\ |Dim(c')| = |Dim(c)| + 1\}$ where $|Dim(c)|$ denotes the cardinality of the set $Dim(c)$. In order to illustrate our work, we shall use the following example throughout the paper.

Example 1. *Consider the fact table Sales in Figure 1(a). Sales has three dimension attributes:* Product, Customer *and* Manufacturer, *and one measure attribute* nbSales. *Figure 1(b) represents its associated data cube. Each node represents a cuboid labeled with its dimensions (for readability, each dimension is designated by its first letter) and its size (number of tuples). The first tuple of*

Sales			
(P)roduct	*(C)ustomer*	*(M)anufacturer*	*nbSales*
P_1	C_1	M_1	*50*
P_2	C_1	M_1	*30*
P_3	C_2	M_1	*40*

(a) FactTable



(b) Cube

Fig. 1. Running example

Sales *means that customer* C_1 *bought 50 units of product* P_1 *and this product is manufactured by* M_1.

4 Selecting Total Cuboids

In this section we address the problem of selecting the cuboids to be materialized in order to speed up queries evaluation. The queries we consider are of the

oarieli@mta.ac.il

form `SELECT * FROM c` where `c` is a cuboid in *Cube*. Since these queries return cuboids, in this section we will not distinguish between queries and cuboids.

Definition 1 (Cost model). *Let $S \subseteq Cube$ be a set of materialized cuboids and q be a query. $\mathcal{C}(q, S)$ denotes the cost q w.r.t S and $\mathcal{C}(q, S) = min_{w \in A_q \cap S}|w|$ where $|w|$ denotes the number of tuples in w. If $A_q \cap S = \emptyset$ then $\mathcal{C}(q, S) = \infty$.*

Intuitively, for evaluating q we use its smallest materialized ancestor. Even if this cost model seems simplistic it has been shown in [20,24] that it reflects actual query costs. Indeed, when using hashing the `GROUP BY` queries evaluation is linear according to the size of the underlying scanned table [16]. The minimal cost of q is obtained when q itself (*i.e.,* its cuboid) is materialized. Below, we define the performance by which a set of materialized cuboids answers some query q. It is simply the ratio between its minimal cost and its actual cost according to the available stored cuboids.

Definition 2 (Performance factor). *Let q be a query and $S \subseteq Cube$ be the set of materialized cuboids. We define $PF(q, S) = \mathcal{C}(q, S)/|q|$.*

Example 1 (Continued). *Suppose that $S = \{PCM, PC, M\}$ and $q =$*`SELECT * FROM C`*. The materialized ancestors of C are PCM and PC. Both have a size equal to 3. Hence we can use either PC or PCM to evaluate q. Since the minimal cost of q corresponds to the size of its corresponding cuboid C and since $|C| = 2$ then $PF(q, S) = 3/2$. Intuitively, this means that by using S the evaluation of q takes 1.5 more time than the minimal time.*

Now we formally define the problem of selecting the cuboids to be materialized under query performance constraint.

Problem statement: Let *Cube* be a data cube and $f \geq 1$ be a performance factor fixed by the user. We want to select the smallest set of cuboids S, in terms of size, such that for each query q, $PF(q, S) \leq f$. In other words, we want each query to be evaluated in a time less than f times its minimal evaluation time.

Definition 3 (f_Ancestor). *Let c and c' be two cuboids such that $c' \in A_c$. c' is an f_ancestor of c if and only if $\frac{|c'|}{|c|} \leq f$.*

We can see that in order to respect the fixed performance factor, each query should be evaluated from one of its f_ancestors.

Example 1 (Continued). *CM is a 2_ancestor of M while PCM and PM are both 3_ancestors of it.*

Definition 4 (f_Closed cuboid). *A cuboid c is f_closed iff $\forall c' \in A_c : |c'| > f \times |c|$. In other words, all ancestors of c have a size at least f times higher than the size of c.*

Example 1 (Continued). *Cuboid M is 1_closed since all of its ancestors have a size strictly greater than $1 \times |M|$. PM is not 1_closed because it has an ancestor (PCM) whose size is less than $1 \times |PM|$.*

oarieli@mta.ac.il

Definition 5 (Functional dependencies (FD)). *Let r be an instance of a relation R defined over a set of attributes D. Let X and Y be subsets of D and $|X|$ denote the size of $\pi_X(r)$ (projection of r on X). We say that r satisfies the FD $X \to Y$ with confidence $c \leq 1$ iff $\frac{|X|}{|XY|} = c$. We note $r \models X \to Y(c)$.*
The FD $X \to Y$ is said exact if $c = 1$, otherwise it is approximate.

Actually, the confidence of an FD $X \to Y(c)$ reflects the average number of Y values associated to each X value. Hence, one can easily see that it is linked to the performance factor introduced before. Indeed, if $r \models X \to Y(c)$ then $PF(X, \{XY\}) = \frac{1}{c}$.

Example 1 (Continued). *Sales $\models P \to C(1)$ which means that to each product corresponds exactly one customer. On another hand, Sales $\models C \to P(2/3)$ which means that a every customer bought in average $3/2$ products.*

Definition 6 (f_Dependencies). *Let T be a fact table and D be its dimensions. Let $X \subseteq D, A \in D$ and $f \geq 1$ be real number. Then $\mathcal{F}_f$ denotes the set of FD's $X \to A(c)$ satisfied by T where $c \geq 1/f$.*

Example 1 (Continued). *One can easily verify that Sales $\models \rho_1 = P \to C(1)$ and Sales $\models \rho_2 = M \to P(1/3)$. Hence, $\rho_1 \in \mathcal{F}_2$ but $\rho_2 \notin \mathcal{F}_2$.*

The next definition extends the classical definition of attributes set closure under exact FD's to the case of approximate FD's.

Definition 7 (f_Closure of a set of attributes). *Let X be a set of dimensions. The f_closure of X w.r.t. $\mathcal{F}_f$, noted X_f^+ is the set of all attributes A such that $X \to A(c) \in \mathcal{F}_f$.*

Example 1 (Continued). *$P_1^+ = PCM$, $C_2^+ = PCM$ and $M_1^+ = M$.*

The following lemma relating closed cuboids to attributes closure is a straightforward result from the above definitions.

Lemma 1. *Let c be a cuboid defined over the dimensions X. Then c is f_closed iff $X_f^+ = X$.*

Proof. $\Rightarrow$: if c on dimensions X is f_closed then, by definition, $\forall Y \in A_c$ we have $|Y| > f \times |X|$. So $\frac{|Y|}{|X|} > f$ and $X \not\to Y(1/f)$ and consequently $X_f^+ = X$.
$\Leftarrow$: if $X_f^+ = X$, then $\nexists A \mid A \notin X$ and $X \to A(1/f)$ is satisfied. Thus, $\frac{|XA|}{|X|} > f$ and so, $|XA| > f \times |X|$. We have X is f_closed by definition. □

Definition 8 (f_Optimal solution). *Let $S \subseteq Cube$. S is f_correct iff for each query q, $PF(q, S) \leq f$. Let $|S| = \sum_{c \in S} |c|$ denote the size of S. Then S is f_optimal iff it is f_correct and there is no other f_correct $S' \subseteq Cube$ such that $|S'| < |S|$.*

oarieli@mta.ac.il

The following proposition shows that the 1_optimal solution is unique and it corresponds to the set of 1_closed cuboids.

Proposition 1. *Let F_1 be the set of 1_closed cuboids and S be a 1_optimal solution. Then $S = F_1$.*

Proof. Suppose that S is an 1_optimal solution. First we show that (i) $F_1 \subseteq S$: Let $c \in F_1$ and $c \notin S$. This means that there does not exist a cuboid $c' \in A_c$ such that $|c| = |c'|$ and this contradicts the fact that S is f_correct. So c must belong to S. In the opposite side (ii) $S \subseteq F_1$: Let $c \in S$ and $c \notin F_1$. Let $X = Dim(c)$ then, by hypothesis, $X_1^+ = Y \supset X$. Let $c' \in A_c$ be such that $Dim(c') = Y$ and $c' \in F_1$. Hence $c' \in S$. Let $S' = S \setminus \{c\}$. Clearly, all queries that are answered from c can be answered from S' by using c' with the same performance factor. Therefore, S is not 1_optimal if it contain c and this makes the contradiction. □

Example 1 (Continued). *$S = \{PCM, CM, M\}$ is a 1_optimal solution and $|S| = 6$. One can easily verify that each of the 8 queries can be evaluated from a cuboid in S whose size is equal to its size.*

Algorithm 1 allows us to find the 1_optimal solution S by computing the 1_closed cuboids. This algorithm is quite naïve since it tests the 1_closedness of all the 2^n cuboids. Presently we are not aware of any algorithm for efficiently computing the 1_closed cuboids. The concept of closed data cube has already been used in the literature, *e.g.,* [10,34] but in a different meaning. Those works consider the tuples values lattice instead of the cuboids lattice. A brute force adaptation of those algorithms would be to check whether all the tuples belonging to a cuboid are closed and if so, we can conclude that the cuboid is closed. It is not clear if this adaptation is efficient enough. Another way to tackle this problem would be to first extract the exact FD's using for example algorithms such that Tane [21] or Fun [28] then computing the closed sets of attributes by exploiting Amstrong axioms. We however note that even if the closure of a set of attributes can be computed in linear time w.r.t the number of FD's, to the best of our knowledge, there is no algorithm capable of computing all the closed sets of attributes in linear or at least in polynomial time depending on the number of such cuboids. Hence, for the moment, we leave for future work the problem of efficiently computing 1_closed cuboids.

While the 1_optimal solution is unique, there may exist several f_optimal solutions. However we can bind the f_optimal solutions by showing that they range between the 1_closed and f_closed cuboids. Before proving this result, we first start by showing that with some lemmas.

Lemma 2. *$\forall f, f'$ such that $f' \leq f$, we have $F_f \subseteq F_{f'}$ where F_f (resp. $F_{f'}$) is the set of f_closed cuboids (resp. f'_closed cuboids).*

Proof. Let c be an f_closed cuboid. By definition, $\forall c' \in A_c$, we have $|c'| > f \times |c|$. Since $f' \leq f$, we can deduce that $|c'| > f \times |c| \geq f' \times |c|$. Hence, c is f'_closed too. □

oarieli@mta.ac.il

Algorithm 1. 1_closed cuboids selection

```
S ← T
for all c ∈ Cube do
    closed(c) ← true
    for all c′ ∈ Cube ∩ P_c and closed(c) = true do
        if |c| = |c′| then
            closed(c) ← false
        end if
    end for
    if closed(c) = true then
        S ← S ∪ c
    end if
end for
return S
```

The following lemma shows the inclusion relationship between an f_optimal and the 1_optimal solutions and so between an f_optimal solution and the set of 1_closed cuboids F_1.

Lemma 3. *Let S_1 be the 1_optimal solution. For each f_optimal solution S we have $S \subseteq S_1$.*

Proof. Let $c \in S$ and suppose that $c \notin S_1$. Let $Dim(c)_1^+ = Dim(c')$ be the closure of c. Let Q_c^f denote the queries we can answer from c with a performance factor less or equal than f (*i.e.,* c is an f_ancestor of all cuboids in Q_c^f). We first show that $Q_c^f \subset Q_{c'}^f$. Indeed, $q \in Q_c^f \Rightarrow q \preceq c \wedge |q| * f \leq |c|$. Obviously $q \preceq c \Rightarrow q \preceq c'$ and $|c| = |c'|$ since $c \notin S_1$. We conclude that $q \in Q_{c'}^f$.

Assume $c' \in S$. In this case, S is not optimal because $S \setminus \{c\}$ can answer the whole queries with the exactly the same performance while it needs strictly less memory.

Let $c' \notin S$. So there exists a set of materialized cuboids $K \subseteq S$ such that $Q_{c'}^f \subset Q_K^f$ otherwise S is not f_correct. Hence $Q_c^f \subset Q_{c'}^f \subset Q_K^f$. Therefore we conclude that S is not optimal since $S \setminus \{c\}$ is a strictly smaller set and answer all the queries with the same performance. □

So far, we have seen that for a 1_optimal solution we materialize all 1_closed cuboids. So, one may wonder whether we need to materialize all f_closed cuboids when we want an f_optimal solution. The following lemma shows that f_closed cuboids are actually part of the solution.

Lemma 4. *If S is an f_optimal solution then $F_f \subseteq S$.*

Proof. With an f_optimal solution S, each query has a cost at most f times the minimal cost. By definition, if c is an f_closed cuboid then it has no f_ancestor. Hence, in order to not violate the performance constraint, c has to be materialized. □

oarieli@mta.ac.il

Now we can prove Proposition 2 which is a direct consequence of the above lemmas.

Proposition 2. *Let S be an f_optimal solution. Then $F_f \subseteq S \subseteq F_1$.*

Example 1 (Continued). *Let $f = 2$. We have $F_1 = \{PCM, CM, M\}$ and $F_2 = \{PCM\}$. PCM alone cannot answer* efficiently *(*i.e., *with a performance $f = 2$) the queries on M and $\emptyset$. Hence, we have to select among $F_1 \setminus F_2 = \{M, CM\}$ the smallest set of cuboids. Since M satisfies the condition and give the smallest set, we conclude that the final solution is $S_2 = \{PCM, M\}$.*

Due to space limitation we will not go further by providing exact or approximate algorithms for finding an f_optimal solution. Note however that one has to choose among the cuboids in $F_1 \setminus F_f$ and this may reduce drastically the search space. Moreover, we can restrict the target queries to be optimized to those that cannot already be answered efficiently from the elements in F_f. The choice can be done either by a greedy algorithm with an approximation guarantee or exactly if the search space is not too large.

In this section the solution consists in either materializing a whole cuboid or not. The next section proposes an extension by allowing some cuboids to be partially materialized.

5 Selecting Partial Cuboids

In this section we consider more general queries by allowing them to contain a `WHERE` clause. The conditions we consider are conjunctions and/or disjunctions of $Dim_i = value$. We relax the constraint saying that a cuboid is either fully materialized or not. Indeed, in order to cope with these queries and respect the performance factor, we have to tolerate that some cuboids are partially materialized.

Example 1 (Continued). *Let $f = 1.5$. The optimal solution with complete cuboids for our example is $S = \{PCM, M\}$. Now let q =*`SELECT * FROM CM WHERE` *$C = C_1$. To evaluate q we have to scan all the tuples*[2] *in PCM where $C = C_1$ hence we have a cost equal to 2 which is greater than f. So, materializing complete cuboids does not allow to answer efficiently this kind of queries.*

Let us first give some notations. The domain of each dimension attribute Dim_i is extended with the new special value *all* such that each tuple τ of each cuboid c can be represented by an *extended tuple* obtained by assigning the value *all* to each $Dim_i \notin Dim(c)$, *e.g.,* the tuple $\langle C_1 \rangle$ of cuboid C is represented by its extension $\langle all, C_1, all \rangle$. The *universe* $\mathcal{U}(Cube)$ of a data cube $Cube$, or simply $\mathcal{U}$, is the set of extended tuples of its cuboids tuples. Now we define a partial order $\trianglelefteq$ between elements of $\mathcal{U}$ as follows: Let $\tau_1 = \langle \tau_{1_1}, \ldots, \tau_{1_n} \rangle$ and $\tau_2 = \langle \tau_{2_1}, \ldots, \tau_{2_n} \rangle$ be two tuples of $\mathcal{U}$ where τ_{i_j} denotes the value of dimension Dim_j in tuple τ_i.

[2] We suppose that indexes are available.

oarieli@mta.ac.il

Then $\tau_1 \trianglelefteq \tau_2$ iff $\forall 1 \leq j \leq n$ either $\tau_{1_j} = \tau_{2_j}$ or $\tau_{1_j} = all$. Clearly, $(\mathcal{U}, \trianglelefteq)$ defines a lattice. For an extended tuple τ, we use D_τ, A_τ and P_τ to denote respectively the descendants, ancestors and parents of τ.

Example 1 (Continued). *Figure 2 represents the tuples lattice of our example. Let* $\tau = \langle P_1, all, all \rangle$, i.e., *the first tuple of cuboid* P. *Then* $D_\tau = \{\langle all, all, all \rangle\}$, $P_\tau = \{\langle P_1, C_1, all \rangle, \langle P_1, all, M_1 \rangle\}$ *and* $A_\tau = P_\tau \cup \{\langle P_1, C_1, M_1 \rangle\} \cup \{\tau\}$. *We deliberately ignore the measure attribute for readability.*

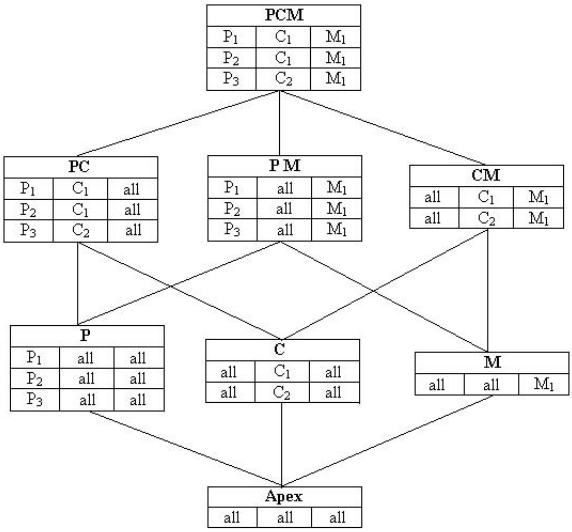


Fig. 2. The cube lattice resulting from *Sales* with extended tuples

In the sequel we consider only extended tuples. Moreover, we will need to refer to the ancestors of a tuple τ belonging to the same cuboid. Let τ be a tuple and X be a set of dimensions such that $Dim(\tau) \subseteq X$. Then $A_\tau(X) = \{\tau' \in A_\tau \mid Dim(\tau') = X\}$. $A_\tau(X)$ is called the X_ancestors of τ, *e.g.*, let $\tau = \langle all, all, M_1 \rangle$ then $A_\tau(CM) = \{\langle all, C_1, M_1 \rangle, \langle all, C_2, M_1 \rangle\}$. The generalization to a set of tuples $\mathcal{T}$ gives: $A_\mathcal{T}(X) = \cup_{\tau \in \mathcal{T}} A_\tau(X)$. If $\mathcal{T}$ is the set of all tuples in a cuboid c, then $A_\mathcal{T}(c' \in A_c)$ is the set of all tuples in c'. The elements of $\mathcal{T}$ are required to belong to the same cuboid otherwise $A_\mathcal{T}$ is not defined. We now extend the f_ancestor definition to tuples.

Definition 9 (f_X_Ancestor of a tuple). *Let* τ *be a tuple. Then the set of ancestors of* τ *in* X, $A_\tau(X) = \mathcal{T}$ *is an* f_X_*Ancestor of* τ *iff* $|\mathcal{T}| \leq f$.

Example 1 (Continued). *Let* $\tau = \langle all, all, M_1 \rangle$. $A_\tau(CM)$ *contains two tuples (see Figure 2). Hence* $A_\tau(CM)$ *is a* $2_CM_ancestor$ *of* τ. $A_\tau(PM)$ *is not* $2_CM_ancestor$ *of* τ *since* $|A_\tau(PM)| = 3$.

oarieli@mta.ac.il

Definition 10 (f_Answering set of a tuple). *Let S be the set of materialized tuples and $\mathcal{T} \subseteq S$. Let $\tau \in \mathcal{U}$. $\mathcal{T}$ is an answering set of τ iff $\bigcup_{\tau' \in \mathcal{T}} A_{\tau'}(T) = A_\tau(T)$ where T is the fact table. In other words, τ can be correctly computed from $\mathcal{T}$*[3]*. If $|\mathcal{T}| \leq f$, we say that $\mathcal{T}$ is an f_answering set of τ.*

Example 2. *By asking for the total number of sold products provided by manufacturer M_1 we obtain the tuple $\tau = \langle all, all, M_1 \rangle$ (we again omit the measure attribute). Then $\mathcal{T} = \{\tau_1, \tau_2\} = \{\langle all, C_1, M_1 \rangle, \langle P_3, all, M_1 \rangle\}$ is a 2_answering set of τ since $A_{\tau_1}(Sales) \cup A_{\tau_2}(Sales) = \{\langle P_1, C_1, M_1 \rangle, \langle P_2, C_1, M_1 \rangle\} \cup \{\langle P_3, C_2, M_1 \rangle\}$ $= A_\tau(Sales)$. On another hand, PCM is a 3_answering set of τ.*

Note that $\mathcal{T}' = \{\tau' = \langle P_2, all, M_1 \rangle, \tau'' = \langle all, C_1, M_1 \rangle\}$ is not an answering set for τ since $\langle P_3, C_2, M_1 \rangle \in A_\tau(Sales)$ and $\langle P_3, C_2, M_1 \rangle \notin A_{\tau'}(Sales) \cup A_{\tau''}(Sales)$.

In the following we define the cost of a query w.r.t a set of materialized tuples S.

Definition 11 (Cost model). *Let $S \subseteq \mathcal{U}$ be the set of materialized tuples.*[4] *The cost of q w.r.t S is the size of its smallest answering set belong to S. More formally $\mathcal{C}(q, S) = \min_{\mathcal{T} \subseteq A_q \cap S \text{ and } A_{\mathcal{T}}(T) = A_q(T)} |\mathcal{T}|$.*

Thus, the minimal cost of a query q, noted $|q|$, is exactly the number of its result tuples.

Example 2 (Continued). *Suppose that $\mathcal{T} \subseteq S$ and $\tau \notin S$. $\mathcal{T}$ is the smallest answering set of τ. Hence, $\mathcal{C}(\tau, S) = 2$.*

Now, we can generalize the definitions seen above to use them more precisely for a tuple.

Definition 12 (f_Closed tuple). *A tuple τ is f_closed iff $\forall X \supseteq Dim(\tau)$: $|A_\tau(X)| > f$. In other words, $\nexists \mathcal{T}$ such that $\mathcal{T}$ is an f_X_ancestor of τ.*

Definition 13 (Conditional functional dependencies). *Consider a relation instance r of a schema R defined over a set of attributes D. Let X and Y be such that $X, Y \subseteq D$. Let $\tau \in \pi_X(r)$. We say r satisfies the conditional functional dependency (CFD) $X_\tau \rightarrow Y$ with a confidence $c \leq 1$ iff $\frac{1}{|A_\tau(XY)|} = c$. We note $r \models X_\tau \rightarrow Y(c)$.*

Example 2 (Continued). *Let $\tau = \langle all, C_1, all \rangle$. Since $A_\tau(PCS)$ contains two tuples which are $\{\langle P_1, C_1, M_1 \rangle$, and $\langle P_2, C_1, M_1 \rangle\}$ then $C_\tau \rightarrow PCS(0.5)$.*

Let $\tau' = \langle all, C_2, all \rangle$, then $C_{\tau'} \rightarrow PCS(1)$. Note that τ is 1_closed while τ' is not since it has a 1_PCS_ancestor.

[3] Since, now a query return a tuple or a set of tuples, we can identify it,like previously, by its result.

[4] We assume that the tuples of the fact table T are always materialized. Hence $T \subseteq S$.

oarieli@mta.ac.il

Definition 14 (f_Closure of a set of tuples). *Let $f \geq 1$. Let $\mathcal{F}_f$ be the set of all CFD's satisfied by the fact table T whose respective confidences are greater than $1/f$. Let τ be a tuple in a cuboid c with $Dim(c) = X$. The f_closure of τ, noted $[X_\tau]_f^+$, is the set of attributes A such that $X_\tau \rightarrow A(c) \in \mathcal{F}_f$.*

In order to distinguish between the f_optimal solutions found in the previous section and those with possibly partial cuboids, we denote the first one by S and the second one by $\mathcal{S}$.

Definition 15 (f_Optimal solution). *A solution $\mathcal{S} \subseteq \mathcal{U}$ is f_optimal iff :*

1. *$\forall q$, $\mathcal{C}(q, \mathcal{S}) \leq f \times |q|$. In other words, $\exists\, \mathcal{T} \in \mathcal{S}$ such that $\mathcal{T}$ is an f_answering set of q.*
2. *$\mathcal{S}$ is the smallest selection between all subsets which satisfy constraint 1 above.*

Concerning the special case of 1_optimal solutions, we have the same result as previously, *i.e.,* the solution is unique and corresponds to the set of 1_closed tuples.

Proposition 3. *Let F_1 be the set of 1_closed tuples and $\mathcal{S}$ be a 1_optimal solution. We have $\mathcal{S} = F_1$.*

Proof. First we show that $F_1 \subseteq \mathcal{S}$: Let $\tau \notin \mathcal{S}$ and $\tau \in F_1$. By definition, τ has no 1_answering set. Therefore we cannot evaluate τ from $\mathcal{S}$ with a cost equal to 1. Which makes $\mathcal{S}$ not a 1_correct solution.

We now prove $\mathcal{S} \subseteq F_1$. Let $\tau \in \mathcal{S}$ and $\tau \notin F_1$. By hypothesis, τ admits at least one 1_answering set $\mathcal{T} = \{\tau'\}$. Either τ' is materialized and then $\mathcal{S}$ is not minimal since $\mathcal{S} \setminus \{\tau\}$ has a lesser size and may answer the same queries as $\mathcal{S}$ does with the same performance. Or τ' is not materialized then it is computed from a materialized 1_ancestor τ'' which is also a 1_ancestor of τ. Again this shows that $\mathcal{S}$ is not minimal. □

Example 2 (Continued). *The 1_closed tuples are $\langle all, C_1, M_1\rangle, \langle all, all, M_1\rangle$ and obviously those of PCM. Note that $|\mathcal{S}| = 5$ whereas the 1_optimal solution when total cuboids have to be selected is $S = \{PCM, CM, M\}$ and $|S| = 6$. By allowing partial cuboids we can further reduce the total size.*

Now we exhibit the relationship existing between the 1_optimal solution S with total cuboids and the 1_optimal solution $\mathcal{S}$ with partial tuples.

Proposition 4. *Let $\mathcal{S}$ be the 1_optimal solution found with CFD's and S be the 1_optimal solution found with just FD's. Then $\mathcal{S} \subseteq S$.*

Proof. Let $\tau \notin S$ be a tuple in a cuboid c. We know that $S = F_1$, so c is not 1_closed. By definition, $\exists c' \in A_c$ s.t. $|c| = |c'|$. Hence, $\exists\{\tau'\} = A_\tau(c')$ such that τ' is a 1_answering set of τ and by definition, τ is not 1_closed. Thus $\tau \notin \mathcal{S}$. □

Surprisingly enough, the above inclusion is not necessary true for $f > 1$. Indeed, it may happen that $\mathcal{S}$ be larger than S.

oarieli@mta.ac.il

Example 2 (Continued). *We have seen previously that* $|S_{1.5}| = 4$. *One can easily check that* $\mathcal{S}_{1.5} = \{\langle all, C_1, M_1\rangle\} \cup S_{1.5}$ *and so* $|\mathcal{S}| = 5$.

The following proposition allows us to characterize a part of an f_optimal solution.

Proposition 5. *Let* $\mathcal{S}$ *be an* f*_optimal solution, then* $F_f \subseteq \mathcal{S}$.

Proof. Let τ be an f_closed tuple ($\tau \in F_f$). τ is necessary materialized since it has no f_answering set. □

By Lemma 5 below, we can reduce the search space of the solution to 1_closed tuples in order to speed up calculations since we can find, for all f_optimal solutions containing no 1_closed tuples, an equivalent solution $\mathcal{S} \subseteq F_1$. We leave open the whether every f_optimal solution $\mathcal{S}$ is included in F_1.

Lemma 5. *Let* $\mathcal{S}'$ *be an* f*_optimal solution. Let* $\tau' \in \mathcal{S}'$ *and* $\tau' \notin F_1$. *Then* $\mathcal{S} = \mathcal{S}' \setminus \{\tau'\} \cup [\tau']_1^+$ *is an* f*_optimal solution.*

Proof. Let $\tau' \in \mathcal{S}'$ be a tuple such that $\tau' \notin F_1$ and $\tau \in F_1$ the 1_closure of τ'. Since $\mathcal{S}'$ is f_optimal, $\tau \notin \mathcal{S}'$. So if we materialize τ instead of τ' in a solution $\mathcal{S}$, then we have obviously $|\mathcal{S}'| = |\mathcal{S}|$.

Let q be a query which uses τ', we have $\mathcal{C}(q, \mathcal{S}') = \mathcal{C}(q, \mathcal{S})$. Thus, we still have a performance guarantee for queries which makes $\mathcal{S}$ f_optimal. □

In the next two sections, we will explain the fact that two reference methods in the field of summarization using FD's to calculate a condensed cube [33] and a quotient cube [23]. Hence, we will see the way to express their solutions with our formalism.

6 Condensed Cubes and CFD's

The condensed cube concept has been proposed in [33]. Intuitively, it consists in removing some tuples from the total data cube when they can be *recovered* from the base tuples. They define a condensed cube as *"a fully pre-computed cube without compression, and, hence, it requires neither decompression nor further aggregation when answering"*. Hence, we can answer all queries by the same way that if we had the complete cube. With our notations, we can say that a condensed cube is a 1_correct solution, so we fix, in this part, $f = 1$.

In order to formally describe this concept, we use an alternative but simpler definition than that of [33].

Definition 16. *Let* T *be a fact table and* $Cube$ *its associated datacube. We suppose that* $Cube$ *is represented by the universe* $\mathcal{U}$. *A condensed cube* CC *is obtained from* $Cube$ *by (i) adding a column* $SDSET$ *to* $Cube$ *where sets of attributes are associated to base tuples and (ii) removing some non base tuples. A non base tuple* τ_1 *s.t* $Dim(\tau_1) = X$ *can be removed from* $Cube$ *if there exists* τ_2 *a base ancestor of* τ_1 *such that*

oarieli@mta.ac.il

1. *$SDSET(\tau_2) \ni X$ and*
2. *τ_2 is the unique base ancestor of τ_1.*

CC is minimal iff there exists no CC' a condensed cube of Cube such that $|CC'| < |CC|$.

The authors show that there exists a unique minimal condensed cube modulo some variations in the $SDSET$ associated to each base tuple.

Example 1 (Continued). *Figure 3(a) represents the datacube when the aggregation measure is SUM. Figure 3(b) represents a minimal condensed cube MCC of Cube together with our 1_optimal solution.*

*From the complete cube, we construct a minimal condensed cube (Figure 3(b)). We can see that all tuples in cuboids P, PC and PM are removed and tuples 14 and 15 corresponding to $C = C_2$ too (*i.e., $\{\{P\}, \{PC\}, \{PM\}\} \subset SDSET()$ *for all base tuples and $\{\{C\}, \{CM\}\} \subset SDSET()$ for the base tuple 3). This means that we can retrieve a tuple in P from Sales.*

From the definition above, we see that a necessary condition for a non base tuple τ to be not present in a condensed cube is that τ admits a single ancestor τ' in the fact table. This means that we have a CFD between τ and the whole set of attributes. We formalize this observation as follows.

Lemma 6. *Let CC be a condensed cube of Cube, $D = Dim(Cube)$ and $\tau \in \mathcal{U} \setminus CC$. Then $[X_\tau]^+ = D$ with $X = Dim(\tau)$.*

Proof. The proof is immediate. It suffices to recall the definition of the tuple closure. □

Note that since $[X_\tau]^+ = D$ and since $D \to M$ (M is the measure attribute) holds in the fact table, by transitivity we conclude that the measure associated to τ is the same as that associated to its unique base ancestor τ'. Thus, the measure of τ can be directly recovered from τ'.

Actually, in order to obtain the minimal condensed cube, it suffices to remove all non base tuples τ such that $[X_\tau]^+ = D$ with $X \subset D$.

Proposition 6. *Let MCC be a minimal condensed cube of Cube and F_1 be the set of 1_closed tuples of Cube. Then $F_1 \subseteq MCC$.*

Proof. Let $\tau \in \mathcal{U}$ and $\tau \notin MCC$. We know that τ has a 1_ancestor in the fact table so τ is not 1_closed, *i.e.*, $\tau \notin F_1$. □

Recall that F_1 is our 1_optimal solution. The main difference between our approach and the condensed cubes resides in the fact that we consider the relations existing between non base tuples while [33] consider only the relations between non base and base tuples. Indeed, in some cases a non base tuple can be recovered from another non base tuple which can be useful in further reducing the data cube.

oarieli@mta.ac.il

TID	Cuboid	Product	Client	Manu-facturer	nbSales
1	Apex	all	all	all	120
2	PCM	P_1	C_1	M_1	50
3	P	P_1	all	all	50
4	PC	P_1	C_1	all	50
5	PM	P_1	all	M_1	50
6	PCM	P_2	C_1	M_1	30
7	P	P_2	all	all	30
8	PC	P_2	C_1	all	30
9	PM	P_2	all	M_1	30
10	PCM	P_3	C_2	M_1	40
11	P	P_3	all	all	40
12	PC	P_3	C_2	all	40
13	PM	P_3	all	M_1	40
14	C	all	C_2	all	40
15	CM	all	C_2	M_1	40
16	C	all	C_1	all	80
17	CM	all	C_1	M_1	80
18	M	all	all	M_1	120

(a) Complete cube

TID	P	C	M	nb	SDSET
1	P_1	C_1	M_1	50	{{P}, {PC}, {PM}, {PCM}}
2	P_2	C_1	M_1	30	{{P}, {PC}, {PM}, {PCM}}
3	P_3	C_2	M_1	40	{{P}, {C}, {PC}, {PM}, {CM}, {PCM}}
4	all	C_1	M_1	80	{}
5	all	C_1	all	80	{}
6	all	all	M_1	120	{}

A minimal condensed cube

TID	P	C	M	nb	SDSET
1	P_1	C_1	M_1	50	{{P}, {PC}, {PM}, {PCM}}
2	P_2	C_1	M_1	30	{{P}, {PC}, {PM}, {PCM}}
3	P_3	C_2	M_1	40	{{P}, {C}, {PC}, {PM}, {CM}, {PCM}}
4	all	C_1	M_1	80	{{C}, {CM}}
5	all	all	M_1	120	{}

Our 1_optimal solution

(b) Solutions

Fig. 3. Comparison of our solution and a minimal condensed cube

Example 1 (Continued). *Figure 3(b) depicts our solution with the same formalism as that for minimal condensed cubes. We note that tuple 5 in MCC has been removed because it can be recovered from tuple 4: it is its unique ancestor among the non base tuples. We moreover add the set* $\{\{C\}, \{CM\}\}$ *to the* $SDSET()$ *of the tuple 4.*

In this section we showed that CFD's are able to express the condensed cube concept. Furthermore, CFD's were helpful in explaining why our solution is also a *lossless summarization* technique while being more effective (cf. Prop. 6).

7 Quotient Cube and CFD's

The concept of Quotient Cube has been presented in [23]. The authors consider a cube lattice in which all cells are tuples. Intuitively, the quotient cube partitions

oarieli@mta.ac.il

those cells into equivalence classes w.r.t the measure associated to them. Each part is *summarized* by its upper and lower bounds. To answer a query, one just needs to know in which class the result belongs to. In other words, let π be class, π^u its upper bound and π^ℓ its lower bound (these are formally defined below), then a tuple τ belongs to π iff there exist $\tau' \in \pi^u$ and $\tau'' \in \pi^\ell$ such that $\tau'' \trianglelefteq \tau \trianglelefteq \tau'$. It turns out that with monotonic aggregate functions (*e.g.,* `COUNT, SUM, MIN, MAX` with positive values) the thus partition itself forms a lattice by considering the pre-order $\sqsubset$ defined as follows: $\pi_1 \sqsubset \pi_2$ iff $\exists \tau_1 \in \pi_1, \tau_2 \in \pi_2$ such that $\tau_1 \trianglelefteq \tau_2$. Let us now give a formal definition of quotient cubes when only monotone aggregate functions are used[5].

Definition 17. *Let $f(\tau)$ denote the measure value associated to τ. Let τ_1 and τ_2 be two tuples in $\mathcal{U}$. τ_1 and τ_2 belong to the same class (*i.e., *τ_1 and τ_2 are equivalent, noted $\tau_1 \equiv \tau_2$) iff $\exists \tau$ such that $\tau_1 \trianglelefteq \tau$, $\tau_2 \trianglelefteq \tau$ and $f(\tau) = f(\tau_1) = f(\tau_2)$.*

Let $\pi = \{\pi_1, \ldots, \pi_p\}$ be the partition induced by $\equiv$ and $\pi_i \in \pi$. The lower and upper bounds of π_i are defined as $\pi_i^\ell = \{\tau \in \pi_i \mid \nexists \tau' \in \pi_i : \tau' \trianglelefteq \tau\}$ and $\pi_i^u = \{\tau \in \pi_i \mid \nexists \tau' \in \pi_i : \tau \trianglelefteq \tau'\}$.

The quotient cube is obtained by storing for each i the bounds π_i^ℓ and π_i^u.

Example 1 (Continued). *Consider our running example with the aggregate function being* `SUM`*. The obtained partition is depicted in Figure 4. There are 5 classes numbered* **I** *to* **V** *plus the idle class* **False***. Each node represents a class labeled by its upper and lower bounds as well as the measure value. The tuple $\langle P_1, C_1, all\rangle$ belongs to class* **I** *because it is bounded by $\langle P_1, all, all\rangle \in \mathbf{I}^\ell$ and $\langle P_1, C_1, M_1\rangle \in \mathbf{I}^u$. On another hand say, class* **III** *is below class* **I** *because there is a tuple $\tau_1 = \langle all, C_1, M_1\rangle \in$* **III** *which is a descendant of $\tau_2 = \langle P_1, C_1, M_1\rangle \in$* **I***.*

We can see that the 1_optimal solution (cf. Figure 3(b)) coincides with the set of upper bounds of the quotient cube.

Indeed, if a tuple is 1_closed, with the aggregate function `SUM` or `COUNT`, then it has not the same measure value as its ancestors so it is the upper bound of a class.

Proposition 7. *Let τ be a tuple in a cuboid X and τ' be a tuple in X'. τ and τ' belong to the same class iff $[X_\tau]^+ = [X'_{\tau'}]^+$.*

Proof. ($\Rightarrow$) If $\tau \equiv \tau'$ then, by definition, it exist a common ancestor t such that $f(t) = f(\tau) = f(\tau')$ (t may be equal to τ or τ'). So τ and τ' can be computed from t and t is in the closure of τ and τ'. However, τ and τ' can not be computed from an ancestor tuple in a such that $[X_\tau]^+ \preceq a$ and $[X'_{\tau'}]^+ \preceq a$, so $[X_\tau]^+$ is the upper bound of the class in which τ belong to and $[X'_{\tau'}]^+$ is the upper bound of the class in which τ' belong to. Since, t belongs to only one class and $t \in [X_\tau]^+ \wedge t \in [X'_{\tau'}]^+$, we deduce that $[X_\tau]^+ = [X'_{\tau'}]^+$.

($\Leftarrow$) Let $t = [X_\tau]^+ = [X'_{\tau'}]^+$ be a tuple. We have $f(t) = f(\tau) = f(\tau')$ and t is a common ancestor to τ and τ'. So τ and τ' belong to the same class. $\square$

[5] The general definition is a bit more complex. However the one given here is sufficient for the functions we consider.

oarieli@mta.ac.il

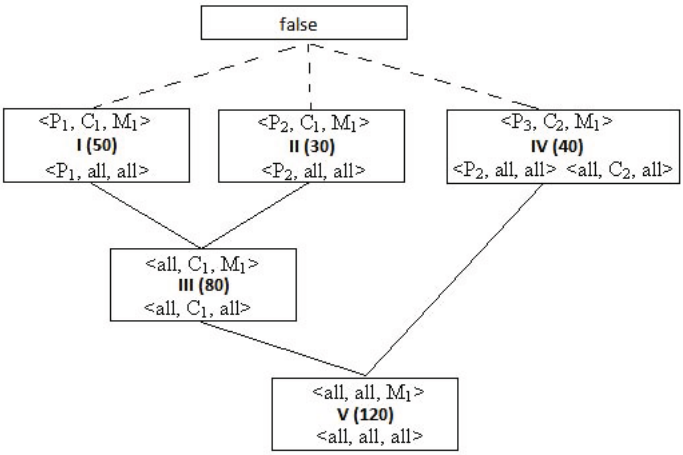


Fig. 4. Quotient Cube with `SUM`

The correspondence between class upper bounds and 1_optimal solutions is not guaranteed for all aggregate functions. Indeed, the following example shows a case where this does not hold.

Example 1 (Continued). *For the aggregate function* `MIN`*, the obtained quotient cube is depicted in Figure 5. This example also shows that depending on the chosen aggregate function, we may obtain different quotient cubes from a single fact table.*

In order to cope with this context, we will redefine the tuples form for which we calculate the closure. Indeed, so far we considered just the dimensions part of each tuple when computing its closure. Since for quotient cubes the measure is important, the tuples will be assumed to contain the measure attribute. Doing so, one can check that the upper bounds of the quotient cube in Figure 5 will coincide with the closed tuples in this new extension. Nevertheless, this will

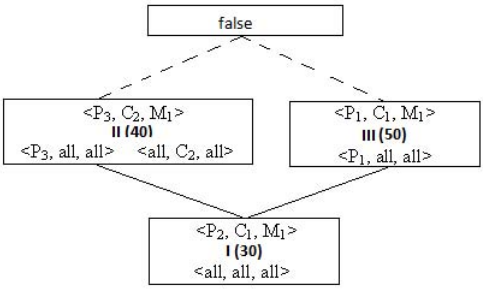


Fig. 5. Quotient Cube with `MIN`

oarieli@mta.ac.il

not always be the case. In fact, the Proposition 7 will hold, for the aggregate function `MIN`, `MAX` or `AVG`, when the upper bounds contain exactly one tuple. It is the reason why we should have a different measure value for each base tuples for those functions.

Our problem is somewhat different since we do not give a particular importance to the aggregated function (like for the condensed cube) whereas we obtain different quotient cubes based on the chosen function. But we can note that, in particular cases, we can identify the upper bound of each class with our notations.

8 Conclusion

Functional dependencies and their variations have been successfully used in relational, nested relational and object oriented databases for reducing redundancy as well as query optimization. In this paper, we investigated their usage in the data cubes context. For this purpose, we used exact and approximate FD's as well as CFD's. We showed that our proposal is effective in two ways: guarantees w.r.t queries evaluation and by demonstrating how previous summarization techniques such as condensed and quotient cubes can be expressed in terms of CFD's. This helped us to show weaknesses and strengths of the method with same concepts. Our characterizations introduce several issues for future works. First, from an algorithmic point of view, we still missing constructive procedures to efficiently compute our solutions, *e.g.,* we want to identify precisely and efficiently which 1_closed cuboids (or tuples) belong to an f_optimal solution. Indeed, even if 1_closed cuboids (tuples) may reduce significantly the search space, their number may still be quite large to use an exact algorithm. Hence, we are investigating approximate solutions with lower complexity. Second, we plan to consider a distributed setting like in [3,32] in order to generalize our characterization by adapting the definitions, *e.g.,* the query cost does not depend solely on the size of the scanned data but on its transfer time too. Another issue will be to propose a concrete implementation of our approach by using efficient data structures to effectively store and query our solutions. We think that the techniques proposed for the condensed cubes may be helpful in this context. Furthermore, even if the objective of [27] is somehow different from ours (sparsity reduction), we are wondering whether it is not possible to establish some equivalence between our and their solutions. This would help us to provide a MOLAP implementation for our work. In the summarization part, we were not able to characterize quotient cubes in the most general setting. We are wondering whether other kinds of dependencies, *e.g.,* multivalued could be helpful. Finally, expressing other summarization techniques, *e.g.,* [9,30], in terms of FD's is another open issue for our future research.

Acknowledgments. We are grateful to the anonymous referees who helped us improving the readability of our paper. We especially thank one referee who pointed out to us reference [27].

oarieli@mta.ac.il

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Baralis, E., Paraboschi, S., Teniente, E.: Materialized view selection in a multidimensional database. In: Proceedings of VLDB Conference, pp. 156–165 (1997)
3. Bauer, A., Lehner, W.: On solving the view selection problem in distributed data warehouse architectures. In: Proceedings of SSDBM Conference, pp. 43–54. IEEE Computer Society (2003)
4. Beeri, C., Bernstein, P.A., Goodman, N.: A sophisticate's introduction to database normalization theory. In: Proceedings of VLDB Conference, pp. 113–124 (1978)
5. Blakeley, J.A., Larson, P.-Å., Tompa, F.W.: Efficiently updating materialized views. In: Proceedings of SIGMOD Conference, pp. 61–71. ACM (1986)
6. Bra, P.D., Paredaens, J.: Conditional Dependencies for Horizontal Decompositions. In: Díaz, J. (ed.) ICALP 1983. LNCS, vol. 154, pp. 67–82. Springer, Heidelberg (1983)
7. Bravo, L., Fan, W., Ma, S.: Extending dependencies with conditions. In: Proceedings of VLDB Conference (2007)
8. Bruno, N.: Automated Physical Database Design and Tuning. CRC Press Inc. (2011)
9. Casali, A., Cicchetti, R., Lakhal, L.: Extracting semantics from data cubes using cube transversals and closures. In: Proceedings of ACM KDD Conference, pp. 69–78. ACM (2003)
10. Casali, A., Nedjar, S., Cicchetti, R., Lakhal, L.: Closed cube lattices. In: New Trends in Data Warehousing and Data Analysis. Annals of Information Systems, vol. 3, pp. 1–20. Springer, Heidelberg (2009)
11. Chaudhuri, S., Lee, H., Narasayya, V.R.: Variance aware optimization of parameterized queries. In: Proceedings of SIGMOD Conference (2010)
12. Chiang, F., Miller, R.J.: Discovering data quality rules. In: Proceedings of VLDB Conference, vol. 1(1), pp. 1166–1177 (2008)
13. Codd, E.F.: Normalized data base structure: A brief tutorial. IBM Research Report, San Jose, California, RJ935 (1971)
14. Codd, E.F.: A relational model of data for large shared data banks (reprint). Communications of ACM 26(1), 64–69 (1983)
15. Golab, L., Karloff, H.J., Korn, F., Srivastava, D., Yu, B.: On generating near-optimal tableaux for conditional functional dependencies. In: Proceedings VLDB Conference, vol. 1(1), pp. 376–390 (2008)
16. Graefe, G.: Query evaluation techniques for large databases. ACM Computing Surveys 25(2), 73–170 (1993)
17. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. Data Mining and Knowledge Discovery 1(1), 29–53 (1997)
18. Hanusse, N., Maabout, S., Tofan, R.: A view selection algorithm with performance guarantee. In: Proceedings of EDBT Conference. ACM International Conference Proceeding Series, vol. 360, pp. 946–957. ACM (2009)
19. Hanusse, N., Maabout, S., Tofan, R.: Revisiting the Partial Data Cube Materialization. In: Eder, J., Bielikova, M., Tjoa, A.M. (eds.) ADBIS 2011. LNCS, vol. 6909, pp. 70–83. Springer, Heidelberg (2011)

oarieli@mta.ac.il

20. Harinarayan, V., Rajaraman, A., Ullman, J.: Implementing data cubes efficiently. In: Proceedings of SIGMOD Conference, pp. 205–216. ACM Press (1996)
21. Huhtala, Y., Kärkkäinen, J., Porkka, P., Toivonen, H.: Tane: An efficient algorithm for discovering functional and approximate dependencies. Computer Journal 42(2), 100–111 (1999)
22. Karloff, H., Mihail, M.: On the complexity of the view-selection problem. In: Proceedings of PODS Conference, pp. 167–173. ACM (1999)
23. Lakshmanan, L.V.S., Pei, J., Han, J.: Quotient cube: How to summarize the semantics of a data cube. In: Proceedings of VLDB Conference. VLDB Endowment, pp. 778–789 (2002)
24. Li, J., Talebi, Z., Chirkova, R., Fathi, Y.: A Formal Model for the Problem of View Selection for Aggregate Queries. In: Eder, J., Haav, H.-M., Kalja, A., Penjam, J. (eds.) ADBIS 2005. LNCS, vol. 3631, pp. 125–138. Springer, Heidelberg (2005)
25. Mannila, H., Räihä, K.-J.: Design of Relational Databases. Addison-Wesley (1992)
26. Microsoft, SqL Server: Database Engine Tuning Advisor, msdn.microsoft.com/en-us/library/ms173494.aspx
27. Niemi, T., Nummenmaa, J., Thanisch, P.: Normalising olap cubes for controlling sparsity. Data & Knowledge Engineering 46(3), 317–343 (2003)
28. Novelli, N., Cicchetti, R.: FUN: An Efficient Algorithm for Mining Functional and Embedded Dependencies. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, pp. 189–203. Springer, Heidelberg (2000)
29. Oracle, Oracle SQL Access Advisor, www.oracle-base.com/articles/11g/SQLAccessAdvisor_11gR1.php
30. Saint-Paul, R., Raschia, G., Mouaddib, N.: General purpose database summarization. In: Proceedings of VLDB Conference, pp. 733–744 (2005)
31. Shukla, A., Deshpande, P., Naughton, J.: Materialized view selection for multidimensional datasets. In: Proceedings of VLDB Conference, pp. 488–499 (1998)
32. Shukla, A., Deshpande, P.M., Naughton, J.F.: Materialized View Selection for Multi-cube Data Models. In: Zaniolo, C., Grust, T., Scholl, M.H., Lockemann, P.C. (eds.) EDBT 2000. LNCS, vol. 1777, pp. 269–284. Springer, Heidelberg (2000)
33. Wang, W., Feng, J., Lu, H., Yu, J.X.: Condensed cube: An effective approach to reducing data cube size. In: Proceedings of ICDE Conference, pp. 155–165. IEEE (2002)
34. Xin, D., Shao, Z., Han, J., Liu, H.: C-cubing: Efficient computation of closed cubes by aggregation-based checking. In: Proceedings of ICDE Conference (2006)

oarieli@mta.ac.il

Armstrong Databases and Reasoning for Functional Dependencies and Cardinality Constraints over Partial Bags

Sven Hartmann[1], Henning Köhler[2], Sebastian Link[3], and Bernhard Thalheim[4]

[1] Institut für Informatik, Technische Universität Clausthal, Germany
[2] N-Squared Software, Palmerston North, New Zealand
[3] Department of Computer Science, University of Auckland, New Zealand
[4] Institut für Informatik, Christian-Albrechts-University Kiel, Germany

Abstract. Data dependencies capture meaningful information about an application domain within the target database. The theory of data dependencies is largely a theory over relations. To make data processing more efficient in practice, partial bags are permitted as database instances to accommodate partial and duplicate information. However, data dependencies interact differently over partial bags than over the idealized special case of relations. In this paper, we study the implication problem of the combined class of functional dependencies and cardinality constraints over partial bags. We establish an axiomatic and an algorithmic characterization of the implication problem. These findings have important applications in database design and data processing. Finally, we investigate structural and computational properties of Armstrong databases for the class of data dependencies under consideration. These results can be utilized to consolidate and communicate the understanding of the application domain between different stake-holders of a database.

1 Introduction

Quality database schemata must capture both the structure and semantics of the underlying application domain. Data dependencies are classes of first-order formulae that can model semantically meaningful information in the target database. In the relational model of data, approximately 100 different classes of data dependencies have been studied [24]. Among those, functional dependencies and cardinality constraints represent two classes of data dependencies that are popular in database practice and theory. Cardinality constraints, in particular, have been studied extensively in Chen's Entity-Relationship model. In practice, however, relations represent idealized special cases in which all information is always available and no duplicate information can occur. In relational database management systems, database instances are partial bags. That is, duplicate rows can occur and columns may contain partial information in the form of null marker occurrences, unless they have been specified as `NOT NULL`. In this paper we are concerned with the implication problem of the combined class of functional dependencies, cardinality constraints and `NOT NULL` constraints over

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 164–183, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

partial bags. The implication problem is to decide whether every partial bag that satisfies a given set of data dependencies also satisfies another given data dependency. The problem is essential in database design, and has found numerous applications in almost all data processing tasks. While different classes of data dependencies co-occur in practice, this co-occurrence is often the source for the intractability or even infeasibility of the associated implication problem. It is therefore a challenge to identify combined classes of data dependencies that can be reasoned about effectively and efficiently.

Example 1. Suppose that in designing an information system for a company the team of data engineers has established the following SQL table definition:

```
CREATE TABLE Employment (
    Emp VARCHAR NOT NULL,
    Dept VARCHAR,
    Mgr VARCHAR NOT NULL);
```

Here, employees (*Emp*) work within a department (*Dept*) under a manager (*Mgr*). Null marker occurrences are only permitted in the column *Dept*. As interpretation of the null marker we choose the most primitive one as *no information*, i.e. a total value may not exist or may exist but is currently unknown. The team of data engineers has started to think about the semantics of the application domain. So far, they have acquired the following business rules. Employees can work for at most one department, and departments have at most one manager. Moreover, every employee can be associated with at most 4 combinations of any department and any manager, every manager can be associated with at most 2 combinations of any employee and any department, and every combination of any employee and any manager must be unique. These business rules can be expressed as functional dependencies and cardinality constraints. The team of engineers would like to consult the experts of the application domain to find out whether their current perceptions about the semantics captures all the requirements necessary. In order to validate their own understanding of the application domain and to facilitate the knowledge acquisition from the domain experts the team plans to create test data, in particular.

Example 1 illustrates how quality database designs require a deep understanding of the application domain's semantics. In particular, it is necessary to comprehend the interactions between different classes of data dependencies in the presence of partial and duplicate information. Such an advanced understanding can also lead to more efficient ways of data processing. For example, suppose that we want to retrieve all `distinct` combinations of an employee and a department from the current database instance. Since the business rules above are enforced on all instances, and since the constraint that *every combination of an employee and a department is unique* is implied by these business rules, it follows that the `distinct` clause in our query is superfluous. Query optimizers with built-in reasoning abilities for these constraints can therefore detect such opportunities effectively, and depending on the complexity of the associated implication

oarieli@mta.ac.il

problem, even efficiently. For these reasons, an in-depth investigation of associated implication problems are both challenging and in high demand.

Contributions. So far, the combined class of functional dependencies and cardinality constraints has only been considered over relations, i.e., in the idealized special case where no duplicate rows and no null marker occurrences are permitted. In this paper we make three major contributions. Firstly, we characterize axiomatically the implication problem for the combined class of functional dependencies, cardinality constraints and `NOT NULL` constraints over partial bags. Secondly, we characterize the implication problem also algorithmically. Our results show *how* reasoning about this combined class of constraints over partial bags can be done effectively and efficiently. For our third and final contribution we investigate the concept of Armstrong databases for the combined class under discussion. We establish structural and computational properties of Armstrong tables. In particular, we characterize the structure of Armstrong tables, i.e., provide sufficient and necessary conditions that allow us to test whether a given partial bag is Armstrong with respect to a given set of constraints in this class. This characterization enables us to derive further properties. For example, we characterize for which sets of constraints in this class Armstrong tables exist. We show that the problem of computing Armstrong tables for a given set of constraints in this class is precisely exponential in the size of the given set. Nevertheless, we are able to establish an algorithm that *always* computes an Armstrong table for a given set of constraints whenever it exists and whose number of rows is at most quadratic in the number of rows of a minimum-sized Armstrong table and the number of given constraints.

Organization. We discuss related work in Section 2. Subsequently, we introduce the data model in Section 3, which includes a definition of the syntax and semantics used in this paper. In Section 4 we characterize the implication problem axiomatically and algorithmically. The structural and computational properties of Armstrong tables are established in Section 5. Finally, we conclude in Section 6 where we also comment briefly on future work. Due to space limitations we have moved some of the proofs into the appendix.

2 Related Work

Data dependencies and Armstrong databases have been studied thoroughly in the relational model of data, cf. [1,9]. Dependencies are essential to the design of the target database, the maintenance of the database during its lifetime, and all major data processing tasks [1,26]. Armstrong databases are a useful design aid for data engineers that can help with the consolidation of data dependencies [16], the design of databases [21] and the creation of concise test data [6].

Armstrong [2] established the first axiomatization for functional dependencies. In general, axiomatizations can be applied by designers and administrators to validate the specification of explicit knowledge, to design and fine-tune databases or to optimize queries. An axiomatization ensures that all opportunities of utilizing implicit knowledge have been exploited. An analysis of the completeness

oarieli@mta.ac.il

argument can provide invaluable hints for finding algorithms that efficiently decide the implication problem. The implication problem of functional dependencies can be decided in time linear in the input [8]. For relations, the structural and computational properties of Armstrong relations for the class of functional dependencies are well-studied [4,21]. Cardinality constraints have mostly been investigated in conceptual models under a relational semantics [10,17,19,25], and recently in XML [13,22].

One of the most important extensions of the basic relational model [5] is incomplete information [15]. This is mainly due to the high demand for the correct handling of such information in real-world applications. Approaches to deal with incomplete information comprise incomplete relations, or-relations or fuzzy relations. In this paper we focus on incomplete relations. In the literature many kinds of null makers have been proposed; for example, "missing" or "value unknown at present", "non-existence", "inapplicable", "no information" and "open". Several works on functional dependencies in incomplete relations exist. Levene and Loizou studied classes of functional dependencies with a weak and strong possible world semantics [18]. Atzeni and Morfuni established an axiomatization of functional dependencies in the presence of `NOT NULL` constraints under the "no information" interpretation [3]. In this context, Hartmann and Link established an equivalence of the implication problem for this class of functional dependencies and `NOT NULL` constraints to that of propositional Horn clauses in Cadoli and Schaerf's family of $\mathcal{S}$-3 logics [14]. Both articles consider only instances where functional dependencies subsume uniqueness constraints, but do not consider neither tables with duplicate rows nor cardinality constraints. In [11] structural and computational properties of Armstrong databases have been established for the combined class of functional dependencies and `NOT NULL` constraints. In the present paper, we draw from this body of research and establish fundamental results for the combined class of functional dependencies, cardinality constraints and `NOT NULL` constraints over partial bags.

3 The Data Model

Let $\mathfrak{H} = \{H_1, H_2, \ldots\}$ be a countably infinite set of symbols, called *column headers* or *headers* for short. A *table schema* is a finite non-empty subset T of $\mathfrak{H}$. Each header H of a table schema T is associated with a countably infinite domain $dom(H)$ of the possible values that can occur in column H. To encompass partial information every column can have a null marker, denoted by $\texttt{ni} \in dom(H)$. The intention of `ni` is to mean "no information". We would like to stress that a null marker is different from a domain value. The inclusion of `ni` into the domain is a syntactic convenience.

For header sets X and Y we may write XY for $X \cup Y$. If $X = \{H_1, \ldots, H_m\}$, then we may write $H_1 \cdots H_m$ for X. In particular, we may write simply H to represent the singleton $\{H\}$. A *row* over T (T-row or simply row, if T is understood) is a function $r : T \to \bigcup_{H \in T} dom(H)$ with $r(H) \in dom(H)$ for all $H \in T$. The null marker occurrence $r(H) = \texttt{ni}$ associated with a header H in

oarieli@mta.ac.il

a row r means that there is no information about $r(H)$. That is, $r(H)$ may not exist or $r(H)$ exists but is unknown. For $X \subseteq T$ let $r(X)$ denote the restriction of the row r over T to X. A *table* t over T is a finite multi-set (bag) of rows over T. We sometimes use the phrase *partial bag* to indicate that these bags can contain partial information in the form of null marker occurrences. In this paper, the terms *table* and *partial bag* can be used interchangeably. For a row r over T and a set $X \subseteq T$, r is said to be X-total if for all $H \in X$, $r(H) \neq \texttt{ni}$. Similar, a table t over T is said to be X-total, if every row r of t is X-total. A table t over T is said to be a *total table* if it is T-total.

Following Atzeni and Morfuni [3] a *null-free subschema* (NFS) over the table schema T is a an expression $nfs(T_s)$ where $T_s \subseteq T$. The NFS T_s over T is satisfied by a table t over T, denoted by $\models_t nfs(T_s)$, if and only if t is T_s-total. SQL allows the specification of column headers as `NOT NULL`. NFSs occur in everyday database practice: the set of headers declared `NOT NULL` forms the single NFS over the underlying table schema.

Following Lien [20] a *functional dependency* (FD) over the table schema T is a statement $X \to Y$ where $X, Y \subseteq T$. The FD $X \to Y$ over T is satisfied by a table t over T, denoted by $\models_t X \to Y$, if and only if for all $r_1, r_2 \in t$ the following holds: if $r_1(X) = r_2(X)$ and r_1, r_2 are X-total, then $r_1(Y) = r_2(Y)$. FDs of the form $\emptyset \to Y$ are called *non-standard*, otherwise FDs are called *standard*. The size $|\sigma|$ of an FD $\sigma = X \to Y$ is defined as $|X| + |Y|$.

We now introduce the concept of a cardinality constraint into databases with partial information. Let $\mathbb{N}$ denote the positive integers. A *cardinality constraint* (CC) over the table schema T is a statement $card(X) \leq b$ where $X \subseteq T$ and $b \in \mathbb{N}$. The CC $card(X) \leq b$ over T is satisfied by a table t over T, denoted by $\models_t card(X) \leq b$, if and only if for all $r_1, r_2, \ldots, r_{b+1} \in t$ the following holds: if $\forall i, j \in \{1, \ldots, b+1\}(r_i(X) = r_j(X))$ and $\forall i \in \{1, \ldots, b+1\}(r_i(X)$ is X-total), then $\exists i, j \in \{1, \ldots, b+1\}(r_i = r_j)$. CCs of the form $card(\emptyset) \leq b$ are called *non-standard*, otherwise CCs are called *standard*. CCs subsume the concept of *uniqueness constraints* for the special case where $card(X) \leq 1$. The size $|\sigma|$ of a CC $\sigma = card(X) \leq b$ is defined as $|X| + \log b$.

For a set Σ of constraints over some table schema T, we say that a table t over T *satisfies* Σ, denoted by $\models_t \Sigma$, if t satisfies every element of Σ. If for some $\sigma \in \Sigma$ the table t does not satisfy σ we sometimes say that t violates σ (in which case t also violates Σ) and write $\not\models_t \sigma$ ($\not\models_t \Sigma$). The size $||\Sigma||$ of a set Σ of FDs and CCs is defined as the sum of sizes over all elements of Σ. The cardinality $|\Sigma|$ of a finite set Σ is defined as the number of its elements.

Example 2. The SQL table definition from Example 1 can be captured in our data model as follows. The table schema T = Employment consists of the column headers *Emp*, *Dept* and *Mgr*. The NFS $nfs(T_s)$ is defined by $T_s = \{Emp, Mgr\}$. The set Σ consists of the FDs $Emp \to Dept$ and $Dept \to Mgr$, and the CCs $card(Emp) \leq 4$, $card(Mgr) \leq 2$ and $card(Emp, Mgr) \leq 1$.

For the design, maintenance and applications of a relational database, data dependencies are identified as semantic constraints on the relations which are intended to be instances of the database schema. During the design process or

oarieli@mta.ac.il

lifetime of a database one usually needs to determine further dependencies which are logically implied by the given ones. In line with the literature of database constraints, we restrict our attention to the implication of constraints in some fixed class $\mathcal{C}$: FDs and CCs in the presence of an NFS.

Let T be a table schema, let $nfs(T_s)$ denote an NFS over T, and let $\Sigma \cup \{\varphi\}$ be a set of FDs and CCs over T. We say that Σ *implies* φ in the presence of $nfs(T_s)$, denoted by $\Sigma \models_{T_s} \varphi$, if every T_s-total table t over T that satisfies Σ also satisfies φ. If Σ does not imply φ in the presence of $nfs(T_s)$ we may also write $\Sigma \not\models_{T_s} \varphi$.

The *implication problem for functional dependencies and cardinality constraints in the presence of a null-free subschema* is to decide, given any table schema T, any NFS $nfs(T_s)$ over T, and any set $\Sigma \cup \{\varphi\}$ of FDs and CCs over T, whether $\Sigma \models_{T_s} \varphi$. For the class of FDs and CCs, the sets $\Sigma \cup \{\varphi\}$ over a fixed table schema T are not necessarily always finite. While for a fixed T there are only finitely many FDs, there might be infinitely many CCs by taking arbitrarily large upper bounds $b \in \mathbb{N}$. However, for a fixed $X \subseteq T$ only the least $b \in \mathbb{N}$ that occurs is relevant. Therefore, we assume without loss of generality that they are finite. Note that for FDs and CCs (in the presence of an NFS) it does not matter whether we restrict our tables to those that are finite, i.e., the implication problem coincides with the finite implication problem where only finite tables are considered. For this reason, we will only speak of the implication problem.

For an FD set Σ over a table schema T and an NFS $nfs(T_s)$ over T, let the FD set $\Sigma^*_{T_s} = \{\varphi \mid \Sigma \models_{T_s} \varphi\}$ denote the *semantic closure* of Σ and $nfs(T_s)$, and for a set $X \subseteq T$ let $X^*_{\Sigma,T_s} = \{H \in T \mid \Sigma \models_{T_s} X \to H\}$ denote the *closure* of X under Σ and $nfs(T_s)$. For a set Σ of FDs and CCs over T let $\Sigma[\text{FD}] = \{X \to Y \mid X \to Y \in \Sigma\} \cup \{X \to T \mid card(X) \leq 1 \in \Sigma\}$. For a set $\Sigma \cup \{\varphi\}$ of FDs and CCs, an NFS $nfs(T_s)$, and a set $\mathfrak{R}$ of inference rules let $\Sigma \vdash_{\mathfrak{R}} \varphi$ denote an *inference* of φ from Σ by $\mathfrak{R}$. That is, there is some sequence $\gamma = [\sigma_1, \ldots, \sigma_n]$ of FDs and CCs such that $\sigma_n = \varphi$ and every σ_i is an element of Σ or results from an application of an inference rule in $\mathfrak{R}$ to some elements in $\{\sigma_1, \ldots, \sigma_{i-1}\}$. For a finite set Σ of FDs and CCs let $\Sigma^+_{\mathfrak{R}} = \{\varphi \mid \Sigma \vdash_{\mathfrak{R}} \varphi\}$ denote the *syntactic closure* of Σ under inferences by $\mathfrak{R}$. $\mathfrak{R}$ is said to be *sound* (*complete*) for the implication of FDs and CCs in the presence of an NFS if for every table schema T, for every NFS $nfs(T_s)$ over T and for every set Σ of FDs and CCs over T we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*_{T_s}$ ($\Sigma^*_{T_s} \subseteq \Sigma^+_{\mathfrak{R}}$). The (finite) set $\mathfrak{R}$ is said to be a (finite) *axiomatization* for the implication of FDs and CCs in the presence of an NFS if $\mathfrak{R}$ is both sound and complete for the implication of FDs and CCs in the presence of an NFS.

Example 3. Consider the set Σ with NFS $nfs(T_s)$ over table schema T from Example 2. Then the following are examples of CCs implied by Σ in the presence of $nfs(T_s)$: $card(Dept) \leq 2$ and $card(Emp, Dept) \leq 1$. However, neither the CC $card(Emp) \leq 2$ nor the FD $Emp \to Mgr$ are implied by Σ in the presence of $nfs(T_s)$. Indeed, the T_s-total table

oarieli@mta.ac.il

Table 1. Axiomatization $\mathfrak{S}$ of FDs and CCs in the presence of an NFS

$\dfrac{}{XY \to X}$ (reflexivity)	$\dfrac{X \to YZ}{X \to Y}$ (decomposition)	$\dfrac{X \to Y \quad X \to Z}{X \to YZ}$ (union)
$\dfrac{X \to Y \quad Y \to Z}{X \to Z}\, Y \subseteq XT_s$ (null transitivity)	$\dfrac{card(X) \leq b}{card(X) \leq b+1}$ (weakening)	$\dfrac{card(X) \leq 1}{X \to T}$ (demotion)
$\dfrac{X \to Y \quad card(Y) \leq b}{card(X) \leq b}\, Y \subseteq XT_s$ (null pullback)		

Emp	*Dept*	*Mgr*
Sisyphus	`ni`	Trump
Sisyphus	`ni`	Gates
Sisyphus	`ni`	Jobs

satisfies Σ, but violates $card(Emp) \leq 2$ and $Emp \to Mgr$.

4 Characterizations of the Implication Problem

The first target in our analysis is the establishment of an axiomatization for the implication of FDs and CCs in the presence of an NFS. The insights from the completeness proof will enable us to characterize the implication problem algorithmically, subsequently.

4.1 Axiomatic Characterization

Let $\mathfrak{S}$ denote the set of inference rules from Table 1. It is our goal to show that $\mathfrak{S}$ forms a finite axiomatization. In our proof we will use the result by Atzeni and Morfuni that the set $\mathfrak{M}$, consisting of the reflexivity axiom, and the decomposition, union and null transitivity rule, forms a finite axiomatization for the implication of FDs [3].

Lemma 1. *The weakening, demotion and null pullback rules are sound for the implication of FDs and CCs in the presence of an NFS.* □

Note that the soundness of the reflexivity axiom and the null pullback rule also imply the soundness of the *superset rule* $\dfrac{card(X) \leq b}{card(XY) \leq b}$. Indeed, the trivial FD $XY \to Y$ and the CC $card(X) \leq b$ allow us to infer the CC $card(XY) \leq b$ by an application of the null pullback rule since $Y \subseteq XYT_s$.

oarieli@mta.ac.il

Example 4. Consider the set Σ with NFS $nfs(T_s)$ over table schema T from Example 2. Then the following are examples of inferences from Σ and $nfs(T_s)$ by $\mathfrak{S}$. An application of the null pullback rule to $Dept \to Mgr$, $card(Mgr) \leq 2$, and $Mgr \in T_s$ results in the CC $card(Dept) \leq 2$. That is,

$$\frac{Dept \to Mgr \quad card(Mgr) \leq 2}{card(Dept) \leq 2} \quad .$$

We now outline an inference of $card(Emp, Dept) \leq 1$ from Σ and $nfs(T_s)$ by $\mathfrak{S}$. Applications of the reflexivity axiom result in $Emp,Dept \to Emp$ and $Emp,Dept \to Dept$. An application of the null transitivity rule to $Emp,Dept \to Dept$, and $Dept \to Mgr$ as well as $Dept \in \{Emp,Dept,Mgr\}$ results in the FD $Emp,Dept \to Mgr$. An application of the union rule to $Emp,Dept \to Emp$ and $Emp,Dept \to Mgr$ results in the FD $Emp,Dept \to Emp,Mgr$. Finally, an application of the null pullback rule to $Emp,Dept \to Emp,Mgr$, $card(Emp,Mgr) \leq 1$, and $\{Emp, Mgr\} \subseteq \{Emp,Dept,Mgr\}$ results in $card(Emp,Dept) \leq 1$. The tree

$$\dfrac{\dfrac{\dfrac{}{Emp,Dept \to Emp} \quad \dfrac{\dfrac{}{Emp,Dept \to Dept} \quad Dept \to Mgr}{Emp,Dept \to Mgr}}{Emp,Dept \to Emp,Mgr} \quad card(Emp,Mgr) \leq 1}{card(Emp,\ Dept) \leq 1}$$

illustrates this inference.

Before we turn to the completeness argument, we want to emphasize that any set of FDs alone can never imply any cardinality constraint.

Proposition 1. *Let T be a table schema, $nfs(T_s)$ an NFS, and Σ a set of FDs over T. Then for all cardinality constraints $card(X) \leq b$ over T we have $\Sigma \not\models_{T_s} card(X) \leq b$.*

Proof. Let t denote the table over T that consists of $b+1$ rows which have for every column header of T the same non-null value, i.e., t consists of $b+1$ duplicate total rows. Clearly, t satisfies Σ and $nfs(T_s)$. Since t violates $card(X) \leq b$ it follows that $\Sigma \not\models_{T_s} card(X) \leq b$. □

Corollary 1. *Let T be a table schema. Then the FD $X \to T$ over T does not imply the cardinality constraint $card(X) \leq 1$.* □

For the completeness of $\mathfrak{S}$ the following lemma is central.

Lemma 2. *Let Σ be a set of FDs and CCs, and $nfs(T_s)$ be an NFS over table schema T. Then the following hold:*

1. *$\Sigma \models_{T_s} X \to Y$ if and only if $\Sigma[\mathrm{FD}] \models_{T_s} X \to Y$, and*
2. *$\Sigma \models_{T_s} card(X) \leq b$ if and only if there is some $card(Y) \leq b' \in \Sigma$ such that $b' \leq b$ and $Y \subseteq XT_s \cap X^*_{\Sigma[\mathrm{FD}],T_s}$.* □

oarieli@mta.ac.il

For the second part of Lemma 2 consider the special case where Σ consists of FDs only. Then no cardinality constraint can be implied by Σ in the presence of the NFS, in consistency with Proposition 1.

We have now the means to verify that $\mathfrak{S}$ is a finite axiomatization for the implication of FDs and CCs in the presence of an NFS. Note that $\mathfrak{S}$ is indeed finite, since the rules apply to any given table schema T, any given sets $X, Y, Z, T_s \subseteq T$ of column headers, and any given $b \in \mathbb{N}$. In particular, the weakening rule applies to every given $b \in \mathbb{N}$.

Theorem 1. *The set $\mathfrak{S}$ is a finite axiomatization for the implication of FDs and CCs in the presence of an NFS.*

Proof. The soundness of $\mathfrak{S}$ follows from Lemma 1 and the soundness of the rules in $\mathfrak{M}$, established in previous work [3]. Let $\Sigma \cup \{\varphi\}$ denote a set of FDs and CCs, and $nfs(T_s)$ denote an NFS over table schema T. For the completeness of $\mathfrak{S}$ we need to show that $\Sigma \models_{T_s} \varphi$ implies $\Sigma \vdash_{\mathfrak{S}} \varphi$. We distinguish between two cases.

Firstly, let φ denote the FD $X \to Y$. From $\Sigma \models_{T_s} X \to Y$ we conclude that $\Sigma[\text{FD}] \models_{T_s} X \to Y$ holds by the first part of Lemma 2. The completeness of $\mathfrak{M}$ for the implication of FDs in the presence of an NFS shows that $\Sigma[\text{FD}] \vdash_{\mathfrak{M}} X \to Y$ holds. Since the demotion rule is part of $\mathfrak{S}$ it follows that $\Sigma \vdash_{\mathfrak{S}} \sigma$ holds for every $\sigma \in \Sigma[\text{FD}]$. From $\mathfrak{M} \subseteq \mathfrak{S}$ we therefore conclude that $\Sigma \vdash_{\mathfrak{S}} X \to Y$ holds indeed.

Secondly, let φ denote the CC $card(X) \leq b$. From the second part of Lemma 2 it follows that $\Sigma[\text{FD}] \models_{T_s} X \to Y$, and that $card(Y) \leq b' \in \Sigma$ for some $Y \subseteq XT_s$ and some $b' \leq b$. The first case of this completeness proof shows that $\Sigma \vdash_{\mathfrak{S}} X \to Y$. An application of the null pullback rule yields $\Sigma \vdash_{\mathfrak{S}} card(X) \leq b'$. Finally, applications of the weakening rule result in $\Sigma \vdash_{\mathfrak{S}} card(X) \leq b$. □

4.2 Algorithmic Characterization

In many situations it is not necessary to compute the set of all constraints implied by a given set. Instead, the question is whether a given fixed candidate constraint is implied by the given set of constraints. We will now investigate an algorithmic characterization of the implication problem for the combined class of functional dependencies and cardinality constraints in the presence of an NFS.

Lemma 2 reduces the implication problem for the combined class of FDs and CCs in the presence of an NFS to the implication problem for the class of FDs in the presence of an NFS. Indeed, $\Sigma \models_{T_s} X \to Y$ if and only if $Y \subseteq X^*_{\Sigma[\text{FD}],T_s}$, and $\Sigma \models_{T_s} card(X) \leq b$ if and only if $Y \subseteq X^*_{\Sigma[\text{FD}],T_s}$ for some $card(Y) \leq b' \in \Sigma$ such that $b' \leq b$ and $Y \subseteq XT_s$. Therefore, the implication problem under consideration has been reduced to the computation of the closure $X^*_{\Sigma[\text{FD}],T_s}$ of a given set X of column headers with respect to a given FD set $\Sigma[\text{FD}]$. This, however, has been done in previous work [3]. For reasons of completeness, we re-state the algorithm here.

oarieli@mta.ac.il

Algorithm 2 (NFSClosure(X,Σ,T_s,T))

Input: set X of column headers, FD set Σ, NFS $nfs(T_s)$ over table schema T
Output: closure X^*_{Σ,T_s} of X with respect to Σ and $nfs(T_s)$
Method:
(A0) CLOSURE := X;
(A1) `repeat`
  OLDCLOSURE := CLOSURE;
  `for all` $U \to V \in \Sigma$ `do`
    `if` $U \subseteq$ CLOSURE $\cap XT_s$ `then`
      CLOSURE := CLOSURE $\cup V$;
    `endif`;
  `enddo`;
 `until` OLDCLOSURE = CLOSURE;
(A2) `return` CLOSURE; □

Theorem 3. *The implication problem* $\Sigma \models_{T_s} \varphi$ *over table schema* T *can be decided in time* $\mathcal{O}(|T| \times ||\Sigma||)$. □

Example 5. Consider the set Σ with NFS $nfs(T_s)$ over table schema T from Example 2. We have shown in Example 4 that $\Sigma \models_{T_s} card(Emp, Dept) \leq 1$. Alternatively, we could confirm this fact by using the second part of Lemma 2 and Algorithm 2. Indeed, it is true that $card(Emp, Mgr) \leq 1 \in \Sigma$ and $\{Emp, Mgr\}$ is a subset of the union of $\{Emp, Dept\}$ and $\{Emp, Mgr\}$, as well as a subset of the closure of $\{Emp, Dept\}$ under Σ[FD] and $nfs(T_s)$. In fact, $\{Emp, Dept\}^*_{\Sigma[\mathrm{FD}],T_s} = \{Emp, Dept, Mgr\}$.

5 Armstrong Tables

In this section we explore the concept of *Armstrong databases* for the combined class of FDs, CCs and `NOT NULL` constraints over partial bags. $\mathcal{C}$-Armstrong databases are sample data that perfectly represent the set Σ of constraints from the class $\mathcal{C}$ currently perceived meaningful. Indeed, they satisfy Σ and violate every constraint in $\mathcal{C}$ not implied by Σ. As such, Armstrong databases are an effective means to consolidate and communicate the current perceptions of an application domain's semantics between various stake-holders of the database [11,21]. We will now extend recent results on Armstrong tables for the combined class of FDs and `NOT NULL` constraints over partial bags [11] by the class of cardinality constraints. Note that these results also extend early work on Armstrong relations for the class of FDs, pioneered by Demetrovics, Mannila, Räihä, Beeri, Dowd, Fagin and Statman [4,7,21].

5.1 Central Concepts

In a first step we will fix various notions required to establish results on the structural and computational properties of Armstrong tables. We begin with the concept most central to this section.

oarieli@mta.ac.il

Definition 1. *Let T be a table schema, $nfs(T_s)$ an NFS, and Σ a set of FDs and CCs over T. A table t over T is said to be* Armstrong *for Σ and $nfs(T_s)$ if and only if*

- *for every FD and CC φ over T: t satisfies φ if and only if $\Sigma \models_{T_s} \varphi$, and*
- *for every $nfs(T'_s)$ over T: t satisfies $nfs(T'_s)$ if and only if $T'_s \subseteq T_s$.* □

Example 6. Consider the set Σ with NFS $nfs(T_s)$ over table schema T from Example 2. Then the following table

Emp	*Dept*	*Mgr*
Sisyphus	`ni`	Trump
Sisyphus	`ni`	Gates
Sisyphus	`ni`	Jobs
Sisyphus	`ni`	Zuckerberg
Gödel	Computer Science	Hilbert
Church	Computer Science	Hilbert
Newton	Physics	Gauss
Leibniz	Mathematics	Gauss

is an Armstrong table for Σ and $nfs(T_s)$.

For characterising the structure of Armstrong tables we need different notions of agreement between rows of a table. The different versions are motivated by the potential occurrence of null markers on the one hand, and the different classes of constraints we consider on the other hand. For functional dependencies it suffices to compare all pairs of distinct rows. Cardinality constraints, however, require us to compare any finite number of distinct rows, essentially up to the maximum bound that occurs in the given set of constraints.

Definition 2. *Let T be a table schema, t a table over T, and r_1, r_2 two rows over T. The* agree set *of r_1 and r_2 is defined as $ag(r_1, r_2) = (X, Y)$ where $X = \{H \in T \mid r_1(H) = r_2(H) \wedge r_1(H) \neq \mathtt{ni}\}$, and $Y = \{H \in T \mid r_1(H) = r_2(H)\}$. The* strong agree set *of r_1 and r_2 is defined as $ag^s(r_1, r_2) = X$ where $ag(r_1, r_2) = (X, Y)$. The* weak agree set *of r_1 and r_2 is defined as $ag^w(r_1, r_2) = Y$ where $ag(r_1, r_2) = (X, Y)$. The* agree set *of t is defined as $ag(t) = \{ag(r_1, r_2) \mid r_1, r_2 \in t \wedge r_1 \neq r_2\}$. The* strong agree set *of t is defined as $ag^s(t) = \{X \mid (X, Y) \in ag(t)\}$. The* weak agree set *of t is defined as $ag^w(t) = \{Y \mid (X, Y) \in ag(t)\}$. For $X \in ag^s(t)$ we define $w(X) = \bigcap\{Y \mid (X, Y) \in ag(t)\}$. For every positive integer $b > 1$ we define $ag^s_b(t) = \{\bigcap_{1 \leq i < j \leq b} ag^s(r_i, r_j) \mid \exists r_1, \ldots, r_b \in t(\forall 1 \leq i < j \leq b(r_i \neq r_j))\}$, $ag^s_1(t) = \{T\}$ and $ag^s_\infty(t) = \emptyset$.* □

Example 7. Let t denote the table from Example 6 that is Armstrong for the set Σ and the NFS $nfs(T_s)$ over table schema T from Example 2. Let r_1, r_2 denote the first two rows of t, respectively. Then $ag(r_1, r_2) = (\{Emp\}, \{Emp, Dept\})$, $w(Emp) = \{Emp, Dept\}$ and $ag^s(t) = ag^s_2(t) = \{\emptyset, \{Emp\}, \{Mgr\}, \{Dept, Mgr\}\}$. Furthermore, $ag^s_3(t) = ag^s_4(t) = \{\emptyset, \{Emp\}\}$.

oarieli@mta.ac.il

An Armstrong table must violate all the cardinality constraints not implied by the given set. It suffices, however, for any non-empty set X to violate the cardinality constraint $card(X) \leq b_X - 1$ where b_X denotes the minimum positive integer for which $card(X) \leq b_X$ is implied. Moreover, if there are two cardinality constraints $card(X) \leq b_X$ and $card(Y) \leq b_Y$ such that $b_X = b_Y$ and $Y \subseteq X$, then it suffices to violate $card(X) \leq b_X - 1$. This motivates the following definitions.

Definition 3. *Let T be a table schema, $nfs(T_s)$ an NFS, and Σ a set of FDs and CCs over T. For $\emptyset \neq X \subseteq T$ let*

$$b_X = \begin{cases} \min\{b \in \mathbb{N} \mid \Sigma \models_{T_s} card(X) \leq b\} & , \textit{if } \{b \in \mathbb{N} \mid \Sigma \models_{T_s} card(X) \leq b\} \neq \emptyset \\ \infty & , \textit{else} \end{cases}.$$

The set $dup_{\Sigma,T_s}(T)$ of duplicate sets *is defined as $dup_{\Sigma,T_s}(T) = \{X \subseteq T \mid \forall H \in T - X(b_{XH} < b_X)\}$.* □

Note that by Lemma 2 we have $b_X = \min\{b \mid \exists card(Y) \leq b \in \Sigma \wedge Y \subseteq XT_s \cap X^+_{\Sigma[\mathrm{FD}],T_s}\}$, if $\{b \in \mathbb{N} \mid \Sigma \models_{T_s} card(X) \leq b\} \neq \emptyset$.

Example 8. Consider the set Σ with NFS $nfs(T_s)$ over table schema T from Example 2. Then we have $b_{Emp} = 4$, $b_{Dept} = b_{Mgr} = b_{Dept,Mgr} = 2$, $b_{Emp,Dept} = b_{Emp,Mgr} = b_{Emp,Dept,Mgr} = 1$. Therefore, $dup_{\Sigma,T_s}(T) = \{\{Emp\}, \{Dept, Mgr\}, \{Emp, Dept, Mgr\}\}$.

An Armstrong table must also violate all the functional dependencies not implied by the given set. However, it suffices for each column header H to violate all FDs $X \to H$ where X is maximal with the property that $X \to H$ is not implied. Furthermore, if X is maximal for some H in this sense, $X \in dup_{\Sigma,T_s}(T)$ and X is not maximal for any $H \in T - T_s$, then we can violate $card(X) \leq b_X - 1$ such that for all $H \in T_s - X$, $X \to H$ is also violated. These arguments motivate the following definitions.

Definition 4. *Let Σ be a set of FDs and let $nfs(T_s)$ be an NFS over table schema T. For a column header $H \in T$ we define the maximal sets $max_{\Sigma,T_s}(H)$ of H with respect to Σ and $nfs(T_s)$ as follows:*

$$max_{\Sigma,T_s}(H) := \{\emptyset \neq X \subseteq T \mid \Sigma \not\models_{T_s} X \to H \wedge \forall H' \in T - X(\Sigma \models_{T_s} XH' \to H)\}.$$

The maximal sets *of T with respect to Σ and $nfs(T_s)$ are defined as $max_{\Sigma,T_s}(T) = \bigcup_{H \in T} max_{\Sigma,T_s}(H)$. If Σ and $nfs(T_s)$ are clear from the context we may simply write $max(H)$ and $max(T)$, respectively. Finally, $max^{red}_{\Sigma,T_s}(T) := max_{\Sigma,T_s}(T) - \{X \in dup_{\Sigma,T_s}(T) \mid \forall H \in T - T_s(X \notin max_{\Sigma,T_s}(H))\}$.* □

Example 9. Consider the set Σ with NFS $nfs(T_s)$ over table schema T from Example 2. Recall from Example 8 that $dup_{\Sigma,T_s}(T) = \{\{Emp\}, \{Dept, Mgr\}, \{Emp, Dept, Mgr\}\}$. As maximal sets we compute $max_{\Sigma,T_s}(Emp) = \{\{Dept, Mgr\}\}$, $max_{\Sigma,T_s}(Dept) = \{\{Mgr\}\}$ and $max_{\Sigma,T_s}(Mgr) = \{\{Emp\}\}$. Therefore, $max^{red}_{\Sigma,T_s}(T) = \{\{Mgr\}\}$.

oarieli@mta.ac.il

5.2 Characterization

We are now in a position to establish sufficient and necessary conditions when a given table is Armstrong for a given set Σ of FDs, CCs and an NFS $nfs(T_s)$. This generalises recent work from the special case where Σ consists of FDs only [11]. In turn, that result had generalised a well-known result by Mannila, Räihä, Beeri, Dowd, Fagin and Statman for FDs over total relations [4,21]. In the following theorem, the first (third) condition ensures that all FDs (CCs) not implied by the given set are violated; and the second (fourth) condition ensures that all implied FDs (CCs) are satisfied. The final condition handles the NFS.

Theorem 4. *Let T be a table schema, Σ a set of standard FDs and standard CCs, and $nfs(T_s)$ an NFS over T. Then for all tables t over T, t is an Armstrong table for Σ and $nfs(T_s)$ if and only if all of the following conditions hold:*

1. $\forall H \in T \forall X \in max_{\Sigma,T_s}(H)(X \in ag^s(t) \wedge H \notin w(X))$,
2. $\forall X \in ag^s(t)(X^*_{\Sigma,T_s} \subseteq w(X))$,
3. $\forall X \in dup_{\Sigma,T_s}(T) \exists Z \in ag^s_{b_X}(t)(X \subseteq Z)$,
4. $\forall card(X) \leq b \in \Sigma \forall Z \in ag^s_{b+1}(t)(X \not\subseteq Z)$,
5. $total(t) = T_s$. □

Example 10. The previous examples show that the table t in Example 6 is Armstrong for the set Σ and the NFS $nfs(T_s)$ over table schema T from Example 2. Indeed, the conditions of Theorem 4 are all satisfied by t.

5.3 Computation

For the computation of an Armstrong table, Theorem 4 suggests that the maximal and duplicate sets need to be computed. The computation of the maximal sets with respect to a set of standard FDs and an NFS $nfs(T_s)$ over table schema T has been studied in [11]. For the computation of duplicate sets we now outline an algorithm that is exponential in the size of Σ. While we leave optimizations of this algorithm for future work, we note that there are sets of FDs and there are sets of CCs, respectively, where every Armstrong table for this set is exponential in the size of Σ, cf. Theorem 7.

Let Σ denote a set of standard FDs and standard CCs, $nfs(T_s)$ an NFS, and X a set of column headers over table schema T. We first compute b_X. We start with $b_X := \infty$ and compute $X^*_{\Sigma[\mathrm{FD}],T_s}$ using Algorithm 2. Then, for each $card(Y) \leq b \in \Sigma$ such that $Y \subseteq XT_s \cap X^*_{\Sigma[\mathrm{FD}],T_s}$ and $b < b_X$ we redefine $b_X := b$. Next we compute the duplicate sets $dup_{\Sigma,T_s}(T)$. We start with $dup_{\Sigma,T_s}(T) = \{X \mid \emptyset \neq X \subseteq T\}$. Then for each $X \in dup_{\Sigma,T_s}(T)$ and each $H \in T - X$ such that $b_{XH} = b_X$ we redefine $dup_{\Sigma,T_s}(T) := dup_{\Sigma,T_s}(T) - \{X\}$. Therefore, the time to compute $dup_{\Sigma,T_s}(T)$ and b_X for each $X \in dup_{\Sigma,T_s}(T)$ is in $\mathcal{O}(2^{|T|} \times |T| \times ||\Sigma||)$.

Algorithm 5 shows the computation of an Armstrong table. The first two steps consist of the computations of the duplicate sets X and their associated b_X, and the computation of the maximal sets covered in previous work. In step (A4), the algorithm generates for each duplicate set X a block of b_X rows that satisfies

oarieli@mta.ac.il

$card(X) \leq b_X$, violates $card(X) \leq b_X - 1$, and violates every FD $X \to H$ where $H \in T_s - X$. Note that in this case there cannot be any $H \in T_s - X$ such that $X \to H$ is implied by Σ and $nfs(T_s)$. Otherwise, since X is a duplicate set it would hold that $card(XH) \leq b < b_X$ is an implied cardinality constraint. Due to the soundness of the null pullback rule, $card(X) \leq b < b_X$ would be implied, too. This is a contradiction. In step (A5), the algorithm computes for each maximal set X the set Z of column headers in $T - T_s$ for which X is maximal, and produces two rows which strongly agree on X and disagree on each column header in Z; unless X is also a duplicate set and each column header for which X is maximal is in T_s. Finally, step (A7) introduces null marker occurrences in every column that is not in T_s, unless such a column already features a null marker occurrence.

Algorithm 5 (Armstrong table computation)

Input: set Σ of standard FDs and standard CCs, an NFS $nfs(T_s)$ over table schema T such that $\forall H \in T \exists b \in \mathbb{N}(\Sigma \models_{T_s} card(H) \leq b)$
Output: Armstrong table t for Σ and $nfs(T_s)$
Method: let $c_{H,1}, c_{H,2}, \ldots \in dom(H)$ be distinct
(A0) compute $dup_{\Sigma,T_s}(T)$ by the procedure outlined above;
(A1) compute $\forall H \in T(max_{\Sigma[\mathrm{FD}],T_s}(H))$ by Algorithm 8 in [11];
(A2) $t := \emptyset$;
(A3) $i := 1$;
(A4) `for all` $X \in dup_{\Sigma,T_s}(T)$ where $b_X > 1$ `do`
$t := t \cup \{r_i, \ldots, r_i + b_X - 1\}$ where $\forall j = i, \ldots, i + b_X - 1$ and $\forall H \in T$

$$r_j(H) := \begin{cases} c_{H,i} & \text{, if } H \in X \\ c_{H,j} & \text{, if } H \in T_s - X \\ \texttt{ni} & \text{, else} \end{cases};$$

$i := i + b_X$;
(A5) `for all` $X \in max^{red}_{\Sigma,T_s}(T)$ `do`
$Z := \{H \in T - T_s \mid X \in max_{\Sigma,T_s}(H)\}$;
$t := t \cup \{r_i, r_{i+1}\}$ where $\forall H \in T$

$$r_i(H) := \begin{cases} c_{H,i} & \text{, if } H \in XZT_s \\ \texttt{ni} & \text{, else} \end{cases}; \text{ and}$$

$$r_{i+1}(H) := \begin{cases} c_{H,i} & \text{, if } H \in X \\ c_{H,i+1} & \text{, if } H \in Z(T_s - X) \\ \texttt{ni} & \text{, else} \end{cases};$$

$i := i + 2$;
(A6) $total(t) := \{H \in T \mid \forall r \in t(r(H) \neq \texttt{ni})\}$;
`if` $total(t) - T_s \neq \emptyset$,
`then return` $t := t \cup \{r_i\}$ where for all $H \in T$,

$$r_i(H) := \begin{cases} \texttt{ni} & \text{, if } H \in total(t) - T_s \\ c_{H,i} & \text{, else} \end{cases};$$

`else return` t;
`endif`; □

Algorithm 5 works correctly.

oarieli@mta.ac.il

Theorem 6. *For every input* $(T, \Sigma, nfs(T_s))$*, where* Σ *is a set of standard FDs and standard CCs, and* $nfs(T_s)$ *is an NFS over table schema* T *such that for all* $H \in T$ *there is some* $b \in \mathbb{N}$ *such that* $\Sigma \models_{T_s} card(H) \leq b$*, Algorithm 5 computes an Armstrong table for* Σ *and* $nfs(T_s)$*.* □

Corollary 2. *Let* Σ *be a set of standard FDs and CCs, and* $nfs(T_s)$ *an NFS over table schema* T*. Then there is a table over* T *that is Armstrong for* Σ *and* $nfs(T_s)$ *if and only if for all* $H \in T$ *there is some* $b \in \mathbb{N}$ *such that* $\Sigma \models_{T_s} card(H) \leq b$*.*

Proof. We show first that the conditions is necessary for the existence of some Armstrong table. Assume, to the contrary, that there is some $H \in T$ such that for all $b \in \mathbb{N}$ we have $\Sigma \not\models_{T_s} card(H) \leq b$. Then there is some $H \in T$ such that $b_H = \infty$. Note that $dup_{\Sigma,T_s}(T) \neq \emptyset$ since $T \in dup_{\Sigma,T_s}(T)$, and $ag^s_\infty(t) = \emptyset$. That is, the third condition of Theorem 4 is always violated. Hence, no Armstrong table over T exists for Σ and $nfs(T_s)$.

The condition is also sufficient. Indeed, under the hypothesis that the condition holds, Algorithm 5 produces an Armstrong table for Σ and $nfs(T_s)$, as verified by Theorem 6. □

Example 11. Consider the set Σ with NFS $nfs(T_s)$ over table schema T from Example 2 as input to Algorithm 5. Then the algorithm generates the following Armstrong table

Emp	*Dept*	*Mgr*
$c_{\text{Emp},1}$	`ni`	$c_{\text{Mgr},1}$
$c_{\text{Emp},1}$	`ni`	$c_{\text{Mgr},2}$
$c_{\text{Emp},1}$	`ni`	$c_{\text{Mgr},3}$
$c_{\text{Emp},1}$	`ni`	$c_{\text{Mgr},4}$
$c_{\text{Emp},5}$	$c_{\text{Dept},5}$	$c_{\text{Mgr},5}$
$c_{\text{Emp},6}$	$c_{\text{Dept},5}$	$c_{\text{Mgr},5}$
$c_{\text{Emp},7}$	$c_{\text{Dept},7}$	$c_{\text{Mgr},7}$
$c_{\text{Emp},8}$	$c_{\text{Dept},8}$	$c_{\text{Mgr},7}$

for Σ and $nfs(T_s)$. Note that after suitable substitutions, this is the Armstrong table given in Example 6.

5.4 Complexity Considerations

Corollary 3. *Let* Σ *be a set of standard FDs and CCs, and* $nfs(T_s)$ *an NFS over table schema* T*. It can be decided in time* $\mathcal{O}(|T|^2 \times ||\Sigma||)$ *whether there is an Armstrong table for* Σ *and* $nfs(T_s)$*.*

Proof. For each $H \in T$ we need to check that there is some $card(X) \leq b \in \Sigma$ such that $X \subseteq HT_s \cap H^*_{\Sigma[\text{FD}],T_s}$, by Lemma 2. This condition can be verified in time $\mathcal{O}(|T| \times ||\Sigma||)$. □

Now we will analyse how well Algorithm 5 does in terms of how well one could potentially do in general. We say that an Armstrong table t for Σ and $nfs(T_s)$

oarieli@mta.ac.il

is said to be *minimum-sized* if there is no Armstrong table t' for Σ and $nfs(T_s)$ such that $|t'| < |t|$.

First of all, the problem of computing an Armstrong table for a given set Σ of standard FDs and standard CCs and an NFS over some table schema is precisely exponential in the size of Σ. If Σ consists of a set of standard FDs only, then this result is known, cf. [11, Proposition 2]. We recall what we mean by *precisely exponential* [4]. Firstly, it means that there is an algorithm for computing an Armstrong table, given a set Σ of standard FDs and standard CCs and an NFS $nfs(T_s)$, where the running time of the algorithm is exponential in $||\Sigma||$. Secondly, it means that there is a set Σ of standard FDs and CCs and an NFS $nfs(T_s)$ in which the number of rows in each minimum-sized Armstrong table for Σ and $nfs(T_s)$ is exponential in $||\Sigma||$ — thus, an exponential amount of time is required in this case simply to write down the table.

Theorem 7. *The problem of computing an Armstrong table for a given set Σ of standard CCs and an NFS* $\mathrm{nfs}(T_s)$ *over table schema T is precisely exponential in the size of Σ.* □

For the remainder of this paper we show that Algorithm 5 is quite conservative in its use of time and space, despite the problem of computing Armstrong tables is computationally hard. We will show that Algorithm 5 *always* computes an Armstrong tables whose number of rows is at most quadratic in the number of rows of a minimum-sized Armstrong table and the cardinality of the given constraint set.

Let Σ denote a set of standard FDs and standard CCs, and $nfs(T_s)$ an NFS over table schema T. We say that a set s_X of rows over T is *X-agreeing* if all rows in s_X strongly agree on X and $|s_X| = b_X$. It follows from Theorem 4 that for every Armstrong table t over T for Σ and $nfs(T_s)$ and every duplicate set $X \in dup_{\Sigma,T_s}(T)$ there is an X-agreeing set $s_X \subseteq t$.

Lemma 3. *Let Σ denote a set of standard FDs and standard CCs, and $nfs(T_s)$ an NFS over table schema T. Let t be a table over T satisfying Σ, and $X, Y \in dup_{\Sigma,T_s}(T)$ with $X \neq Y$ and $b_X = b_Y = b_{X\cap Y}$. Let further s_X, s_Y be X- and Y-agreeing subsets of t, respectively. Then $s_X \cap s_Y = \emptyset$.*

Proof. Assume $r \in s_X \cap s_Y$. Then s_X, s_Y both strongly agree with r on $X \cap Y$, so $s_X \cup s_Y$ strongly agrees on $X \cap Y$. Since $b_X = b_Y = b_{X\cap Y}$ and t satisfies Σ we must have $|s_X \cup s_Y| \leq b_X = b_Y$, and hence $s_X = s_Y$. This in turn implies that $s_X = s_Y$ strongly agrees on $X \cup Y$, and using again that t satisfies Σ we get $b_{X\cup Y} \geq b_X = b_Y$. From $X \neq Y$ it follows that $X \cup Y$ is a proper superset of X and/or Y. Together with $b_{X\cup Y} \geq b_X = b_Y$ this contradicts $X, Y \in dup_{\Sigma,T_s}(T)$. □

Let $card(Y) \leq b \in \Sigma$ and $X \to Y \in \Sigma[\mathrm{FD}]^*_{T_s}$ with $Y \subseteq XT_s$. In particular, $card(X) \leq b$ can be derived using the null-pullback rule. We say that $card(Y) \leq b$ is a *source* of $card(X) \leq b$. For a set $X \subseteq T$ we call $card(Y) \leq b$ a source of X if $card(Y) \leq b$ is a source of $card(X) \leq b_X$.

oarieli@mta.ac.il

Corollary 4. *Let Σ denote a set of standard FDs and standard CCs, and $nfs(T_s)$ an NFS over table schema T. Every cardinality constraint over T of the form $card(X) \leq b_X$ has a source in Σ.*

Proof. By Lemma 2 there is some $card(Y) \leq b' \in \Sigma$ with $b' \leq b_X$ and $Y \subseteq XT_s \cap X^*_{\Sigma[\mathrm{FD}],T_s}$. The latter condition is equivalent to $X \to Y \in \Sigma[\mathrm{FD}]^*_{T_s}$ and $Y \subseteq XT_s$. From this and $card(Y) \leq b' \in \Sigma$ we can derive $card(X) \leq b'$ using the null-pullback rule, and by definition of b_X we have $b' \geq b_X$. This shows $b' = b_X$, so $card(Y) \leq b_X$ is a source of X in Σ. □

We denote by $dup_{\Sigma,T_s}(card(Y) \leq b)$ the set of all duplicate sets for which $card(Y) \leq b$ is a source

$$dup_{\Sigma,T_s}(card(Y) \leq b) := \left\{X \in dup_{\Sigma,T_s}(T) \mid card(Y) \leq b \text{ is a source of } X\right\}.$$

Lemma 4. *Let Σ denote a set of standard FDs and standard CCs, and $nfs(T_s)$ an NFS over table schema T. Let t be an Armstrong table for Σ and $nfs(T_s)$, and $card(Y) \leq b \in \Sigma$. Then $|t| \geq |dup_{\Sigma,T_s}(card(Y) \leq b)| \cdot b$.*

Proof. (1) For each $X \in dup_{\Sigma,T_s}(card(Y) \leq b)$ we have $Y \subseteq XT_s$ and $X \to Y \in \Sigma[\mathrm{FD}]^*_{T_s}$. This implies $XY \subseteq XT_s$ and $X \to XY \in \Sigma[\mathrm{FD}]^*_{T_s}$. By definition of b_{XY} we have $\Sigma \models_{T_s} card(XY) \leq b_{XY}$. Using null-pullback we can derive $card(X) \leq b_{XY}$, so $b_X \leq b_{XY}$. Since X is a duplicate set, we must have $Y \subseteq X$.

(2) Table t contains an X-agreeing set s_X for every $X \in dup_{\Sigma,T_s}(T)$ by Theorem 4, so in particular for every $X \in dup_{\Sigma,T_s}(card(Y) \leq b)$. For every pair of distinct duplicate sets $X_1, X_2 \in dup_{\Sigma,T_s}(card(Y) \leq b)$ we have $Y \subseteq X_1 \cap X_2$ by (1), and hence $b_{X_1} = b_{X_2} = b_Y = b_{X_1 \cap X_2}$. Thus, s_{X_1} and s_{X_2} are disjoint by Lemma 3. This gives us $|dup_{\Sigma,T_s}(card(Y) \leq b)|$ disjoint sets $s_X \subseteq t$, each of which contains b tuples, and shows the bound on $|t|$. □

Corollary 5. *Let Σ denote a set of standard FDs and standard CCs, and $nfs(T_s)$ an NFS over table schema T. Let t be an Armstrong table for Σ and $nfs(T_s)$ over T and $\mathcal{D} := dup_{\Sigma,T_s}(T)$. Then*

$$|t| \geq \frac{\sum_{X \in \mathcal{D}} b_X}{|\Sigma|}.$$

Proof. By Corollary 4 every $X \in \mathcal{D}$ has a source in Σ, so

$$\sum_{card(Y) \leq b \in \Sigma} \left(\sum_{X \in dup_{\Sigma,T_s}(card(Y) \leq b)} b_X \right) \geq \sum_{X \in \mathcal{D}} b_X$$

By Lemma 4, $|t| \geq \sum_{X \in dup_{\Sigma,T_s}(card(Y) \leq b)} b_X$ for any $card(Y) \leq b \in \Sigma$, and thus

$$|t| \cdot |\Sigma| \geq \sum_{card(Y) \leq b \in \Sigma} \left(\sum_{X \in dup_{\Sigma,T_s}(card(Y) \leq b)} b_X \right) \geq \sum_{X \in \mathcal{D}} b_X.$$

□

oarieli@mta.ac.il

Theorem 8. *Let Σ denote a set of standard FDs and standard CCs, and $nfs(T_s)$ an NFS over table schema T. Let t be an Armstrong table for Σ and $nfs(T_s)$ and t_c the Armstrong table for Σ and $nfs(T_s)$ constructed in Algorithm 5. Then $|t_c| \leq |t| \cdot (|t| + |\Sigma|)$.*

Proof. Denote by t_{A4}, t_{A5} and t_{A6} the subsets of t_c constructed in steps (A4), (A5) and (A6) of Algorithm 5, respectively. By Corollary 5 we have

$$|t_{A4}| = \sum_{X \in dup_{\Sigma,T_s}(T)} b_X \leq |t| \cdot |\Sigma|.$$

Steps (A5) and (A6) together construct a sub-table of that computed by Algorithm 10 in [11], thus giving us the bound (Corollary 5 in [11])

$$|t_{A5} \cup t_{A6}| \leq |t|^2.$$

Combining these results yields the theorem. □

Corollary 6. *Algorithm 5 computes an Armstrong table for Σ and $nfs(T_s)$ whose number of rows is at most quadratic in the number of rows of a minimum-sized Armstrong table for Σ and $nfs(T_s)$ and the cardinality of Σ.* □

Finally, we show that, in general, there is no most concise way of representing the information inherent in a set of standard CCs and a null-free subschema. In fact, there are cases in which the size of a minimum-sized Armstrong table is exponential in the size of the constraint set, and there are other cases in which the size of an optimal cover of a constraint set is exponential in the size of a minimum-sized Armstrong table.

Theorem 9. *Let $\mathcal{C}$ denote the class of FDs and CCs. There is some table schema T, some set Σ of CCs and some NFS $nfs(T_s)$ over T such that Σ has size $\mathcal{O}(n)$, and the size of a minimum-sized $\mathcal{C}$-Armstrong table for Σ and $nfs(T_s)$ is $\mathcal{O}(2^n)$. There is some table schema T, some set Σ of CCs and some NFS T_s over T such that there is a $\mathcal{C}$-Armstrong table for Σ and $nfs(T_s)$ where the number of rows is in $\mathcal{O}(n)$, and the optimal cover of Σ with respect to $nfs(T_s)$ has size $\mathcal{O}(2^n)$.*

Proof. Let $T = H_1 \cdots H_{2n}$, $T_s = T$ and let Σ consist of the following standard CCs: for all $i = 1, \ldots, n$, $card(H_{2i-1}H_{2i}) \leq 1$, and for all $i = 1, \ldots, 2n$, $card(H_i) \leq 2$. Then $dup_{\Sigma,T_s}(T)$ contains the 2^n sets $X \subseteq T$ where for each $i = 1, \ldots, n$ either $H_{2i-1} \in X$ or $H_{2i} \in X$. According to Theorem 4 every Armstrong table for Σ and $nfs(T_s)$ contains a number of rows that is exponential in $||\Sigma||$. A similar construction was used in [4] to show that the size of a minimum-sized Armstrong relation can be exponential in the size of a given FD set.

Let $T = H_1H'_1 \cdots H_nH'_n$, $T_s = T$, and let Σ consist of the following standard CCs: for all $i = 1, \ldots, n$, $card(H_i) \leq 3$ and $card(H'_i) \leq 3$, and for all $X = X_1 \cdots X_n$ where $X_i \in \{H_i, H'_i\}$, $card(X) \leq 2$. Then Σ is its own optimal cover,

oarieli@mta.ac.il

i.e. there is no equivalent set Σ' of standard FDs and standard CCs such that $||\Sigma'|| < ||\Sigma||$. The size $||\Sigma||$ is in $\mathcal{O}(2^n)$. Furthermore, $dup_{\Sigma,T_s}(T)$ consists of the n sets $T - H_i H'_i$ for $i = 1, \ldots, n$, and the set T, and $max_{\Sigma,T_s}(T)$ consists of the $2n$ sets $T - H_i$ and $T - H'_i$ for $i = 1, \ldots, n$. Thus, Algorithm 5 computes an Armstrong table for Σ and $nfs(T_s)$ whose number of rows is in $\mathcal{O}(n)$. □

For these reasons we recommend the use of both representations. Indeed, the representation in form of constraint sets enables design teams to identify constraints they currently incorrectly perceive as semantically meaningful; and the representation in form of an Armstrong table enables design teams to identify constraints they currently incorrectly perceive as semantically meaningless.

6 Conclusion and Future Work

We have investigated the combined class of functional dependencies, cardinality constraints and `NOT NULL` constraints over partial bags. This framework applies to the structure of SQL tables. We have characterized the associated implication problem of this class axiomatically and algorithmically. Our results show how reasoning about this expressive class of constraints can be done effectively and efficiently. Moreover, we have established several structural and computational properties of Armstrong tables for this class of constraints. Our results show how Armstrong tables can be used effectively to consolidate the semantics of an application domain expressed by the class of constraints studied.

In future work we would like to implement our algorithms within a design tool. Such a tool may also be used to conduct empirical studies on the usefulness of Armstrong tables for the acquisition of semantically meaningful constraints in our class studied, very much along the lines of [16]. It seems desirable to extend our results to even more expressive classes of constraints, e.g. classes of multivalued and inclusion dependencies. Another challenging problem would be an extension to classes of cardinality constraints that also enforce lower bounds. Properties of Armstrong databases should also be studied in probabilistic and graph databases, and the concept of informative Armstrong databases should be investigated in non-relational models [6]. It would also be interesting to analyse interactions of cardinality constraints and functional dependencies under different interpretations of the null marker [12,18,23].

Acknowledgement. This research is supported by the Marsden fund council from Government funding, administered by the Royal Society of New Zealand.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Armstrong, W.W.: Dependency structures of database relationships. Information Processing 74, 580–583 (1974)

oarieli@mta.ac.il

3. Atzeni, P., Morfuni, N.: Functional dependencies and constraints on null values in database relations. Information and Control 70(1), 1–31 (1986)
4. Beeri, C., Dowd, M., Fagin, R., Statman, R.: On the structure of Armstrong relations for functional dependencies. J. ACM 31(1), 30–46 (1984)
5. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (1970)
6. De Marchi, F., Petit, J.-M.: Semantic sampling of existing databases through informative Armstrong databases. Inf. Syst. 32(3), 446–457 (2007)
7. Demetrovics, J.: On the equivalence of candidate keys with Sperner systems. Acta Cybern. 4, 247–252 (1980)
8. Diederich, J., Milton, J.: New methods and fast algorithms for database normalization. ACM Trans. Database Syst. 13(3), 339–365 (1988)
9. Fagin, R.: Armstrong databases. Technical Report RJ3440(40926), IBM Research Laboratory, San Jose, California, USA (1982)
10. Hartmann, S.: On the implication problem for cardinality constraints and functional dependencies. Ann. Math. Art. Intell. 33, 253–307 (2001)
11. Hartmann, S., Kirchberg, M., Link, S.: Design by example for SQL table definitions with functional dependencies. The VLDB Journal (2011), doi:10.1007/ s00778-011-0239-5
12. Hartmann, S., Leck, U., Link, S.: On Codd families of keys over incomplete relations. The Computer Journal 54(7), 1166–1180 (2011)
13. Hartmann, S., Link, S.: Numerical constraints on XML data. Inf. Comput. 208(5), 521–544 (2010)
14. Hartmann, S., Link, S.: When data dependencies over SQL tables meet the Logics of Paradox and S-3. In: PODS Conference (2010)
15. Imielinski, T., Lipski Jr., W.: Incomplete information in relational databases. J. ACM 31(4), 761–791 (1984)
16. Langeveldt, W.-D., Link, S.: Empirical evidence for the usefulness of Armstrong relations in the acquisition of meaningful functional dependencies. Inf. Syst. 35(3), 352–374 (2010)
17. Lenzerini, M., Nobili, P.: On the satisfiability of dependency constraints in entity-relationship schemata. Inf. Syst. 15(4), 453–461 (1990)
18. Levene, M., Loizou, G.: Axiomatisation of functional dependencies in incomplete relations. Theor. Comput. Sci. 206(1-2), 283–300 (1998)
19. Liddle, S., Embley, D., Woodfield, S.: Cardinality constraints in semantic data models. Data Knowl. Eng. 11, 235–270 (1993)
20. Lien, E.: On the equivalence of database models. J. ACM 29(2), 333–362 (1982)
21. Mannila, H., Räihä, K.-J.: Design by example: An application of Armstrong relations. J. Comput. Syst. Sci. 33(2), 126–141 (1986)
22. Sali, A., Schewe, K.-D.: Keys and Armstrong databases in trees with restructuring. Acta Cybern. 18(3), 529–556 (2008)
23. Thalheim, B.: On semantic issues connected with keys in relational databases permitting null values. Elektronische Informationsverarbeitung und Kybernetik 25(1-2), 11–20 (1989)
24. Thalheim, B.: Dependencies in relational databases. Teubner (1991)
25. Thalheim, B.: Fundamentals of Cardinality Constraints. In: Pernul, G., Tjoa, A.M. (eds.) ER 1992. LNCS, vol. 645, pp. 7–23. Springer, Heidelberg (1992)
26. Thalheim, B.: Entity-Relationship modeling. Springer, Heidelberg (2000)

oarieli@mta.ac.il

FD Covers and Universal Complements of Simple Projections

Stephen J. Hegner

Umeå University, Department of Computing Science
SE-901 87 Umeå, Sweden
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

Abstract. The constant-complement strategy, in which the admissible updates to a given view are those which hold a second complementary view constant, remains one of the most attractive formalisms for identifying suitable translation mechanisms for updates to views of database schemata. However, in general, it suffers from the drawback that the reflections of view updates to the main schema can depend upon the choice of complement in various ways. To overcome this drawback completely, a special kind of complement, called a universal complement, is required. In this paper, sufficient conditions for the existence of such a complement are established for a classical but nevertheless very important setting — views defined by simple projection of a universal relational schema constrained by functional dependencies (FDs). Certain uniqueness properties of covers of these dependencies prove critical in the characterization. The results are extended to quasi-universal complements, which are unique up to exchange of equivalent attributes, thus recapturing certain situations for which unique covers do not exist.

1 Introduction

Although introduced more than thirty years ago, the constant-complement approach to the support of view updates [4] remains an attractive solution formalism. A *complement* of the view Γ is a second view Γ' with the property that the pair $\{\Gamma, \Gamma'\}$ defines a lossless decomposition of the main schema $\mathbf{D}$. In the constant-complement update strategy on Γ for complement Γ', an allowable update u to Γ is precisely one which may be realized as an update on $\mathbf{D}$ in such a way that the state of Γ' does not change. Since $\{\Gamma, \Gamma'\}$ forms a lossless decomposition, there can be at most one such *reflection* of u to an update on $\mathbf{D}$.

This strategy has substantial intuitive appeal because the update to $\mathbf{D}$ is localized — it affects only the part of $\mathbf{D}$ defined by Γ. The "rest" of $\mathbf{D}$, defined by Γ', is unaffected by the update. For this reason, among others, it has seen renewed interest in recent years as a fundamental technique [18] [19] [13], as well as a related principle for other approaches requiring view updates, such as those based upon lenses [9], [11].

In the ideal case, the complement Γ' should possess three distinct properties relative to a class $\mathcal{V}$ of views (such as simple projections) to which it belongs:

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 184–202, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

State invariance: The admissibility of an update u to Γ with constant complement Γ' should be independent of the state of Γ'. This mirrors the idea of a *closed* view, as described in [13], in which the user of the view only needs to know the state of the view itself to determine whether or not an update to it is allowed.

Reflection invariance; If $\Gamma'' \in \mathcal{V}$ is another complement of Γ, and the update u to Γ is also supported by constant complement Γ'', then the reflection of the view update to the main schema should be the same in each case.

Universality: If $\Gamma'' \in \mathcal{V}$ is another complement of Γ, and the update u to Γ is supported by constant-complement Γ'', then u should also be supported by constant complement Γ'.

The first two of these topics have already been studied extensively. State invariance is obtained precisely in the case that the congruences (i.e., the equivalence relations defined by the view mappings) of the two views commute. Basic ideas along these lines were discussed already in [3, Def. 14 and Thm. 14], although a specific characterization in terms of the meet is first presented in [12] and refined greatly in [13], where it is also shown that for the case of projections of relational schemata constrained by usual database dependencies, state invariance is obtained precisely in the case that a cover of the dependencies of the main schema embed into the two views [13, Prop. 2.17].

Reflection invariance is also investigated extensively in [13], where it is shown that order structure is critical. In particular, for the case of relational schemata, if the constraints are usual database dependencies and the views are defined as combinations of select, project, join, and rename, then reflection invariance is assured [13, Prop. 2.5 and Thm. 4.3].

Comparable studies of universality are not to be found in the literature. Formally, a universal complement is one which is least within a given class $\mathcal{V}$ of views, since the smaller the complement, the greater the set of updates supported. For $\mathcal{V}$ the set of projections on a relational schema $\mathbf{E}$, this means that the projection $\Pi^{\mathbf{E}}_{\mathbf{W}'}$ onto attributes $\mathbf{W}'$ is a universal complement of $\Pi^{\mathbf{E}}_{\mathbf{W}}$ if for any other complement $\Pi^{\mathbf{E}}_{\mathbf{W}''}$, $\mathbf{W}' \subseteq \mathbf{W}''$. A study of universality thus amounts to a study of least complements within a given class $\mathcal{V}$. In [15], conditions which guarantee universality for projections are developed in terms of so-called *governing* join dependencies, but they are difficult to translate into more fundamental properties. The primary goal of this paper is to seek a more concrete characterization of universality, in terms of the basic properties of schemata and views. In contrast to [15], the work here requires state invariance as a precondition for all complements. Thus, $\mathcal{V}$ is restricted to complements which exhibit state invariance – the so-called *meet complements*. The reason is that without it, there are simple counterexamples to universality, such as the following, taken from [15]. Let $\mathbf{E}_0$ be the relational schema with the single relation symbol $R[ABCD]$, governed by the functional dependencies (FDs) in $\mathcal{F}_0 = \{B \to D, C \to D\}$. Observe that this schema is completely symmetric with respect to attributes B and C. Consider the view $\Pi^{\mathbf{E}_0}_{ABC}$, the projection onto attributes ABC. The

oarieli@mta.ac.il

projection $\Pi_{CD}^{\mathbf{E}_0}$ onto CD is a minimal complement. Indeed. $ABC \cap CD = C$ is a key for the schema of $\Pi_{CD}^{\mathbf{E}_0}$, and so from the classical result of Rissanen [25, Thm. 1] it follows that the decomposition is lossless. That $\Pi_{CD}^{\mathbf{E}_0}$ is a minimal complement amongst projections is immediate, since neither $\Pi_{C}^{\mathbf{E}_0}$ nor $\Pi_{D}^{\mathbf{E}_0}$ can be complements. On the other hand, given the symmetry in B and C, it follows that $\Pi_{BD}^{\mathbf{E}_0}$ is also a minimal complement of $\Pi_{ABC}^{\mathbf{E}_0}$. Thus, there can be no universal complement amongst the projections of $\mathbf{E}_0$. This implies that there are some updates to $\Pi_{ABC}^{\mathbf{E}_0}$ which are translatable with constant complement $\Pi_{BD}^{\mathbf{E}_0}$ but not with constant complement $\Pi_{CD}^{\mathbf{E}_0}$, and conversely. Specifically, no update which does not keep $\Pi_{B}^{\mathbf{E}_0}$ constant can be realized with constant complement $\Pi_{BD}^{\mathbf{E}_0}$, and no update which does not keep $\Pi_{C}^{\mathbf{E}_0}$ constant can be realized with constant complement $\Pi_{CD}^{\mathbf{E}_0}$, The only updates to $\Pi_{ABC}^{\mathbf{E}_0}$ which keep both complements constant are those in which neither the BD nor the CD projection is allowed to vary. On the other hand, in view of the results of [13], this example does however exhibit reflection invariance — updates to $\Pi_{ABC}^{\mathbf{E}_0}$ which keep both CD and BD constant; i.e., those which change only the values on attribute A, are reflected in the same way with complement $\Pi_{BD}^{\mathbf{E}_0}$ as with $\Pi_{CD}^{\mathbf{E}_0}$.

While it is possible to find examples of situations which exhibit both update-state and reflection invariance, yet deny a universal complement, it is established in this paper that there are reasonable and nontrivial conditions which do guarantee a universal complement. The context is the classical setting of universal relational schemata constrained by FDs, with views defined by simple (i.e., single) projections. The properties which guarantee universality are of covers of the FDs governing the main schema.

The approach is divided into three main phases. In the first, a strong condition is identified which ensures that every projection $\Pi_{\mathbf{W}}^{\mathbf{E}}$ of the universal-relational schema $\mathbf{E}$ has a unique minimal *cover precomplement*; that is, a second projection $\Pi_{\mathbf{W}'}^{\mathbf{E}}$ with the property that a cover of the FDs of $\mathbf{E}$ embed into these views. In the second phase, conditions are established for extension of a such a minimal precomplement to a universal complement. In the third and final phase, it is shown how to relax the conditions of the first two phases to allow for equivalences, and thus include common constructions such as multiple candidate keys. The first two phases are covered in Sec. 3, while the third is presented in 4. Sec. 2 provides background information and Sec. 5 provides conclusions and further directions.

2 Schemata, Views, and Complements

The setting of this work is the classical relational model, with which it is assumed that the reader is familiar. For any necessary clarification, [21] provides perhaps the best presentation. of the necessary context. However, other classical textbooks, such as [24] and [1] should prove more than adequate as well.

The purpose of this section is primarily to present necessary notation and terminology, as well as constructs which build upon that classical model,

oarieli@mta.ac.il

particularly on the subject of views. Thus, this section need only be skimmed on a first reading, and then used as a reference as the need arises.

Summary 2.1 (Attributes and relation schemes). Attributes will normally be written using (possibly subscripted and/or primed) capital letters from the beginning of the alphabet. Thus, A, B_1, and C' denote attributes. Sets of attributes will normally be written using boldface letters from the end of the alphabet; thus $\mathbf{U}$, $\mathbf{W}'$, and $\mathbf{V}_1$ denote sets of attributes. Following the long-standard notation, sets of attributes are represented via concatenation. Thus, ABC is shorthand for $\{A, B, C\}$. In particular, a single attribute name, such as A, represents both attribute itself and the singleton set $\{A\}$. Context will always make clear which is intended.

A *relation scheme* is defined by a name and a nonempty set of attributes. Thus, $R[ABC]$ is a relation scheme. Constraints are not part of a relation scheme.

It is assumed that the reader is familiar with the notions of domain for an attribute and tuple for a relation scheme. These notions will be used only tangentially, and so there is no need to develop a notation carefully.

Summary 2.2 (Functional dependencies). Familiarity with the basic syntax and semantics of functional dependencies (FDs) is assumed. The FD $\mathbf{X} \to \mathbf{Y}$ is *degenerate* if $\mathbf{X} = \emptyset$. In this work, degenerate FDs will not be considered, so the term *FD* should always be taken to mean *nondegenerate FD*. The FD $\mathbf{X} \to \mathbf{Y}$ is *trivial* if $\mathbf{Y} \subseteq \mathbf{X}$. Since trivial FDs always hold, they are not particularly interesting, although they may arise in certain constructions.

$\mathcal{F} \models \varphi$ denotes that the set $\mathcal{F}$ of FDs *entails* or *implies* the FD φ; i.e., φ holds whenever $\mathcal{F}$ holds. For a second set $\mathcal{F}'$ of FDs, write $\mathcal{F} \models \mathcal{F}'$ if $\mathcal{F} \models \varphi$ for every $\varphi \in \mathcal{F}'$. Write $\mathcal{F} \models\!\!\!\dashv \mathcal{F}'$ to denote that both $\mathcal{F} \models \mathcal{F}'$ and $\mathcal{F}' \models \mathcal{F}$ hold, and say that $\mathcal{F}$ and $\mathcal{F}'$ are *equivalent* in this case.

Given an FD $\varphi = \mathbf{X} \to \mathbf{Y}$, define $\mathsf{LHS}(\varphi) = \mathbf{X}$, $\mathsf{RHS}(\varphi) = \mathbf{Y}$, and $\mathsf{Attr}(\varphi) = \mathsf{LHS}(\varphi) \cup \mathsf{RHS}(\varphi)$. This notation is extended to sets of FDs in a natural way: $\mathsf{LHS}(\mathcal{F}) = \{\mathsf{LHS}(\varphi) \mid \varphi \in \mathcal{F}\}$; $\mathsf{RHS}(\mathcal{F}) = \{\mathsf{RHS}(\varphi) \mid \varphi \in \mathcal{F}\}$; $\mathsf{Attr}(\mathcal{F}) = \{\mathsf{Attr}(\varphi) \mid \varphi \in \mathcal{F}\}$. Furthermore, define $\mathsf{Attr}_{\cup}(\mathcal{F}) = \bigcup \mathsf{Attr}(\mathcal{F})$.

The FD φ is *RHS-simple* if $\mathsf{RHS}(\varphi)$ consists of exactly one attribute, and the set $\mathcal{F}$ of FDs is *RHS-simple* if each of its members has that property. φ is *simple* if both $\mathsf{LHS}(\varphi)$ and $\mathsf{RHS}(\varphi)$ consist of exactly one attribute.

Given a set $\mathcal{F}$ of FDs, the *closure* of $\mathcal{F}$, denoted $\mathcal{F}^+$, is the set of all FDs which are implied by those in $\mathcal{F}$. The set $\mathcal{F}$ is *nonredundant* if for any $\mathcal{F}' \subseteq \mathcal{F}$, if $\mathcal{F}'^+ = \mathcal{F}^+$, then $\mathcal{F}' = \mathcal{F}$.

An FD $\mathbf{X} \to \mathbf{Y} \in \mathcal{F}^+$ is *LHS-reduced* or *full* for $\mathcal{F}$ if for any $\mathbf{X}' \subseteq \mathbf{X}$, if $(\mathcal{F} \setminus \{\mathbf{X} \to \mathbf{Y}\}) \cup \{\mathbf{X}' \to \mathbf{Y}\}^+ = \mathcal{F}^+$, then $\mathbf{X}' = \mathbf{X}$. The set $\mathcal{F}$ is *LHS-reduced* if each of its members has that property with respect to $\mathcal{F}$.

If $\mathcal{F}$ is nonredundant, LHS-reduced, and RHS-simple, it is called a *canonical* set of FDs.

A *cover* of $\mathcal{F}$ is any set $\mathcal{C}$ of FDs with the property that $\mathcal{C}^+ = \mathcal{F}^+$; or, equivalently $\mathcal{C} \models\!\!\!\dashv \mathcal{F}$. The set $\mathcal{F}$ is *minimum* if each of its covers contains at least as many FDs as $\mathcal{F}$ itself.

oarieli@mta.ac.il

A *context* is a pair $\langle \mathbf{U}, \mathcal{F} \rangle$ in which $\mathbf{U}$ is a nonempty set of attributes and $\mathcal{F}$ is set of FDs with $\mathsf{Attr}_{\cup}(\mathcal{F}) \subseteq \mathbf{U}$. Notions such as entailment and closure may be used within a context, provided a little care is taken. Formally, define $\mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle) = \mathcal{F} \cup \{A \rightarrow A \mid A \in \mathbf{U} \setminus \mathsf{Attr}_{\cup}(\mathcal{F})\}$, and $\langle \mathbf{U}, \mathcal{F} \rangle^+ = \mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle)^+$. Thus, $\mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle)$ is $\mathcal{F}$ augmented with trivial FDs which involve the attributes of $\mathbf{U}$ not found in any FD of $\mathcal{F}$.

A *superkey* for $\langle \mathbf{U}, \mathcal{F} \rangle$ is a $\mathbf{K} \subseteq \mathbf{U}$ with the property $\mathbf{K} \rightarrow A \in \mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle)^+$ for every $A \in \mathbf{U}$. A minimal superkey (with respect to subset inclusion) is called a *key*. Note that the property of being a superkey or key depends upon both $\mathbf{U}$ and $\mathcal{F}$, and not upon $\mathcal{F}$ alone, since there may be attributes in $\mathbf{U}$ which do not occur in $\mathcal{F}$. However, it is easy to see that $\mathbf{K}$ is a key (resp. superkey) for $\langle \mathbf{U}, \mathcal{F} \rangle$ iff $\mathbf{K} \cap \mathsf{Attr}_{\cup}(\mathcal{F})$ is a key (resp. superkey) for $\langle \mathsf{Attr}_{\cup}(\mathcal{F}), \mathcal{F} \rangle$. In other words, a key (resp. superkey) for $\langle \mathbf{U}, \mathcal{F} \rangle$ may be obtained from one for $\langle \mathsf{Attr}_{\cup}(\mathcal{F}), \mathcal{F} \rangle$ by adding those attributes in $\mathbf{U}$ which do not occur in any FD of $\mathcal{F}$. Thus, a *key* (resp. *superkey*) for $\mathcal{F}$ may and will be defined unambiguously to be a key for $\langle \mathsf{Attr}_{\cup}(\mathcal{F}), \mathcal{F} \rangle$.

If $\mathbf{X} \rightarrow \mathbf{Y} \in \mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle)$ with $\mathbf{X} \subseteq \mathbf{Y}$, it will also be said that $\mathbf{X}$ is a *superkey* for $\mathbf{Y}$.

$\mathbf{X} \leftrightarrow \mathbf{Y}$ is an abbreviation for $\mathbf{X} \rightarrow \mathbf{Y} \wedge \mathbf{Y} \rightarrow \mathbf{X}$, and the statement $\mathbf{X} \leftrightarrow \mathbf{Y} \in \mathcal{F}^+$ is used as a synonym for $\{\mathbf{X} \rightarrow \mathbf{Y}, \mathbf{Y} \rightarrow \mathbf{X}\} \subseteq \mathcal{F}^+$. In this case, it is said that $\mathbf{X}$ and $\mathbf{Y}$ are *equivalent* in $\mathcal{F}$. If both $\mathbf{X}$ and $\mathbf{Y}$ are singletons, so that $\mathbf{X} \leftrightarrow \mathbf{Y}$ is of the form $A \leftrightarrow B$, then it is called a *simple equivalence*.

Summary 2.3 (Relational Schemata and Π-Views). A *universal relational schema* is a pair $\mathbf{E} = (R[\mathbf{U}], \mathsf{Constr}(\mathbf{E}))$ in which in which $R[\mathbf{U}]$ is a relation scheme and $\mathsf{Constr}(\mathbf{E})$ is a set of dependencies on $R[\mathbf{U}]$. $\mathsf{LDB}(\mathbf{E})$ denotes the set of all *legal databases* of $\mathbf{E}$; that is, all sets of tuples on $R[\mathbf{U}]$ which satisfy the constraints in $\mathsf{Constr}(\mathbf{E})$. If $\mathsf{Constr}(\mathbf{E})$ is a set of FDs, then $\mathbf{E}$ is called a *universal FD-schema*, with $\langle \mathbf{U}, \mathcal{F} \rangle$ the *context* of $\mathbf{E}$.

It is worth noting that the universal FD-schema $\mathbf{E} = (R[\mathbf{U}], \mathcal{F})$ and its context $\langle \mathbf{U}, \mathcal{F} \rangle$ determine each other, up to the name of the relation R. However, it nevertheless seems appropriate to maintain this distinction. With $\langle \mathbf{U}, \mathcal{F} \rangle$, the focus is on the dependencies and nothing more, while with $\mathbf{E}$, notions such as $\mathsf{LDB}(\mathbf{E})$ (which involve the relation symbol R) are central.

A *Π-view* of the universal relational schema $\mathbf{E}$ is a pair $\Pi^{\mathbf{E}}_{\mathbf{W}} = (\mathbf{E}^{\mathbf{W}}, \pi^{\mathbf{E}}_{\mathbf{W}})$ in which $\mathbf{E}^{\mathbf{W}} = (R_{\mathbf{W}}[\mathbf{W}], \mathsf{Constr}(\mathbf{E}^{\mathbf{W}}))$ is a relational schema and $\pi^{\mathbf{E}}_{\mathbf{W}}$ is the projection morphism which sends a relation on attribute set $\mathbf{U}$ to its projection onto attribute set $\mathbf{W}$. The set $\mathsf{Constr}(\mathbf{E}^{\mathbf{W}})$ consists of the *implied constraints* [17] on the view schema. The notation $\mathsf{LDB}(\Pi^{\mathbf{E}}_{\mathbf{W}})$ is used as a synonym for $\mathsf{LDB}(\mathbf{E}^{W})$.

Now assume further that $\mathbf{E}$ is a universal FD-schema. It is unfortunately not the case that $\mathbf{E}^{\mathbf{W}}$ will always have that same property. Indeed $\mathsf{Constr}(\mathbf{E}^{\mathbf{W}})$ may not have a finite cover and thus need not consist only of functional dependencies [16, Lem. 4.1], [14, App. A]. Call $\Pi^{\mathbf{E}}_{\mathbf{W}}$ *closed* if $\mathsf{Constr}(\mathbf{E}^{\mathbf{W}})$ is equivalent to a subset of $\mathcal{F}^+$; that is, if the implied constraints on the view have a basis consisting of FDs.

oarieli@mta.ac.il

Fortunately, the views to be considered in this work will always be closed. Specifically, if $\Pi^{\mathbf{E}}_{\mathbf{W}}$ has a meet complement (as defined in Summary 2.4 below), then it is closed [13, Prop. 2.17]. Since the complements constructed in this work will always be meet complements, it follows that all views of universal FD-schemata which are considered in this work will be closed.

There is a natural order[1] on Π-views of $\mathbf{E}$ given by $\Pi^{\mathbf{E}}_{\mathbf{W}_1} \preceq \Pi^{\mathbf{E}}_{\mathbf{W}_2}$ iff $\mathbf{W}_1 \subseteq \mathbf{W}_2$. Similarly, $\Pi^{\mathbf{E}}_{\mathbf{W}_1} \prec \Pi^{\mathbf{E}}_{\mathbf{W}_2}$ iff $\mathbf{W}_1 \subsetneq \mathbf{W}_2$; i.e., $\mathbf{W}_1 \subseteq \mathbf{W}_2$ but $\mathbf{W}_1 \neq \mathbf{W}_2$.

Summary 2.4 (Complementary and covered pairs of views). For a comprehensive presentation of complementary views based upon congruences, see [13] and [14], and for a general characterization of view ordering, also based upon congruences, see [15, Summary 2.1]. In this paragraph, only the ideas necessary to the special setting of this paper are presented, and in some cases, the definitions themselves are recast to capture this special setting only.

Let $\mathbf{E} = (R[\mathbf{U}], \mathcal{F})$ be a universal FD-schema, and let $\Pi^{\mathbf{E}}_{\mathbf{W}_i} = (\mathbf{E}^{\mathbf{W}_i}, \pi^{\mathbf{E}}_{\mathbf{W}_i})$ be Π-views of $\mathbf{E}$ for $i \in \{1, 2\}$. The pair $P = \{\Pi^{\mathbf{E}}_{\mathbf{W}_1}, \Pi^{\mathbf{E}}_{\mathbf{W}_2}\}$ is *complementary* if it is possible to recover the state of $\mathbf{E}$ from the combined states of the two views. In light of the classical result [25, Thm. 1], this condition may be recaptured by defining P to be a *complementary pair* if $\mathbf{W}_1 \cup \mathbf{W}_2 = \mathbf{U}$ and at least one of the FDs $\mathbf{W}_1 \cap \mathbf{W}_2 \to \mathbf{W}_1$, $\mathbf{W}_1 \cap \mathbf{W}_2 \to \mathbf{W}_2$ is in $\mathcal{F}^+$. In this paper, this classical characterization will be used as the definition of a complementary pair.

The members of a complementary pair are called *Π-complements* of each other. As only Π-views are considered in this work, the term *complement* should always be interpreted as Π-complement, unless stated explicitly to the contrary.

The set $\mathcal{F}$ *embeds into* P if every $\varphi \in \mathcal{F}$ embeds into at least one of the views in P (i.e., $\mathsf{Attr}(\varphi) \subseteq \mathbf{W}_i$ for $i = 1$ or $i = 2$.) An *embedded cover* of $\mathcal{F}$ for P is a set $\mathcal{C}$ of FDs which is a cover for $\mathcal{F}$ and each element of $\mathcal{C}$ embeds in P. Call P a *covered pair* if $\mathbf{W}_1 \cup \mathbf{W}_2 = \mathbf{U}$ and there is an embedded cover of $\mathcal{F}$ for P. A covered pair need not be complementary. For example, if $\mathcal{F} = \emptyset$, then P is trivially a covered pair, but it is not a complementary pair unless $\mathbf{W}_1 = \mathbf{U}$ or $\mathbf{W}_2 = \mathbf{U}$.

Call P a *meet-complementary pair* if it is a complementary and covered pair. In this case, the Π-view $\Pi^{\mathbf{E}}_{\mathbf{W}_1 \cap \mathbf{W}_2}$ defined by the attributes common to both $\Pi^{\mathbf{E}}_{\mathbf{W}_1}$ and $\Pi^{\mathbf{E}}_{\mathbf{W}_2}$ is called the *meet* of P. Meet complementary pairs have the following important *independence* property: If $N_1 \in \mathsf{LDB}(\Pi^{\mathbf{E}}_{\mathbf{W}_1})$ and $N_2 \in \mathsf{LDB}(\Pi^{\mathbf{E}}_{\mathbf{W}_2})$ have the property that they agree on the meet $\Pi^{\mathbf{E}}_{\mathbf{W}_1 \cap \mathbf{W}_2}$, then there is a (necessarily unique) $M \in \mathsf{LDB}(\mathbf{E})$ with $\pi^{\mathbf{E}}_{\mathbf{W}_i}(M) = N_i$ for $i \in \{1, 2\}$ [13, Thm. 2.14 and Prop. 2.17].

If P is a complementary pair (resp. covered pair, resp. meet-complementary pair), then $\Pi^{\mathbf{E}}_{\mathbf{W}_1}$ and $\Pi^{\mathbf{E}}_{\mathbf{W}_2}$ are called *complements* (resp. *cover precomplements*, resp. *meet complements*) of each other. Minimality and optimality are defined with respect to the ordering $\preceq$ [10, 1.23]. Specifically, $\Pi^{\mathbf{E}}_{\mathbf{W}_2}$ is a *minimal*

[1] The definition given here is specialized to projections only. For a general definition based upon congruences, see [15, Summary 2.1].

oarieli@mta.ac.il

Π-complement (resp. cover Π-precomplement, resp. meet Π-complement) of $\Pi^{\mathbf{E}}_{\mathbf{W}_1}$ if it is a minimal element with respect to the ordering $\preceq$ when restricted to all complements (resp. cover precomplements, resp. meet complements) of $\Pi^{\mathbf{E}}_{\mathbf{W}_1}$. Similarly, $\Pi^{\mathbf{E}}_{\mathbf{W}_2}$ is an *optimal* Π-complement (resp. cover Π-precomplement, resp. meet Π-complement) of $\Pi^{\mathbf{E}}_{\mathbf{W}_2}$ if it is a least element with respect to the ordering $\preceq$ when restricted to all complements (resp. cover precomplements, resp. meet complements) of $\Pi^{\mathbf{E}}_{\mathbf{W}_1}$. See [15, Summary 2.2] for an elaboration of this idea.

A minimal (resp. optimal) meet Π-complement is also called a *fully minimal* (resp. *fully optimal*) Π-complement. A fully optimal Π-complement satisfies all three conditions identified in the introduction: state-invariance, reflection invariance, and universality, while a fully minimal Π-complement need only satisfy the first two. A fully optimal Π-complement is also called a *universal Π-complement*. Since the smaller the complement, the more updates which are supported via constant complement, a universal Π-complement supports all view updates which any other complement does. Thus, this formalization recaptures the notion of universality identified in Sec. 1, when restricted to meet complements.

These relationships need not be symmetric. For example, let $\mathbf{E}_1$ have a $R[ABCD]$ constrained by the FDs in $\mathcal{F}_1 = \{A \to B, B \to C, C \to D\}$. The universal Π-complement of $\Pi^{\mathbf{E}_1}_{ABC}$ is $\Pi^{\mathbf{E}_1}_{BCD}$, while the universal Π-complement of $\Pi^{\mathbf{E}_1}_{BCD}$ is $\Pi^{\mathbf{E}_1}_{AB}$.

3 Universal Π-Complements and Unique Covers

Notation 3.1 (Notational convention). Unless specifically stated to the contrary, for the rest of this paper, take $\mathbf{E} = (R[\mathbf{U}], \mathcal{F})$ to be a universal FD-schema. In particular, $\mathbf{U}$ is a nonempty set of attributes and $\mathcal{F}$ is a set of FDs over $\mathbf{U}$.

Definition 3.2 (Cover precomplements with respect to fixed set of FDs). Let $\mathbf{W} \subseteq \mathbf{U}$. Suppose that the goal is to identify a cover precomplement $\Pi^{\mathbf{E}}_{\mathbf{W}'}$ of $\Pi^{\mathbf{E}}_{\mathbf{W}}$. The first requirement is that $\mathbf{W} \cup \mathbf{W}' = \mathbf{U}$, and this implies in particular that every attribute of $\mathbf{U}$ which is not in $\mathbf{W}$ must lie in $\mathbf{W}'$. The set of all such attributes is called the *base complement* of $\mathbf{W}$ with respect to $\mathbf{U}$, and it occurs often enough to warrant its own notation.

(a) $\mathsf{BaseCompl}_{\langle \mathbf{U}, \mathcal{F} \rangle}\langle W \rangle = \mathbf{U} \setminus \mathbf{W}$.

The second requirement for a precomplement is that a cover of $\mathcal{F}$ embed into the pair $\{\Pi^{\mathbf{E}}_{\mathbf{W}}, \Pi^{\mathbf{E}}_{\mathbf{W}'}\}$. As a first step, consider the simpler problem of identifying the least $\mathbf{W}'$ with the property that each element of $\mathcal{F}$ embed into the pair. In other words, $\mathcal{F}$ is to be used as its own embedded cover. Although the end result of a solution to that problem is not a goal in itself, the associated construction will prove to be central in the more general case. In this case, all FDs in $\mathcal{F}$ which do not embed into the view $\Pi^{\mathbf{E}}_{\mathbf{W}}$ must embed into $\Pi^{\mathbf{E}}_{\mathbf{W}'}$. This implies that the

oarieli@mta.ac.il

attributes in the *dependency-preserving extension* for $\mathbf{W}$ with respect to $\langle \mathbf{U}, \mathcal{F} \rangle$, defined as follows, must be a subset of $\mathbf{W}'$.

(b) $\mathsf{DPExt}_{\langle \mathbf{U}, \mathcal{F} \rangle} \langle \mathbf{W} \rangle = \bigcup \{ \mathbf{V} \in \mathsf{Attr}(\mathcal{F}) \mid \mathbf{V} \not\subseteq \mathbf{W} \}$.

Putting these two together, the formula for the attributes of $\mathbf{W}'$ is the *precomplement set* of $\mathbf{W}$ with respect to $\langle \mathbf{U}, \mathcal{F} \rangle$, defined as follows.

(c) $\mathsf{PreCompl}_{\langle \mathbf{U}, \mathcal{F} \rangle} \langle \mathbf{W} \rangle = \mathsf{BaseCompl}_{\langle \mathbf{U}, \mathcal{F} \rangle} \langle \mathbf{W} \rangle \cup \mathsf{DPExt}_{\langle \mathbf{U}, \mathcal{F} \rangle} \langle \mathbf{W} \rangle$.

Owing to the importance of this construction, it is recorded as a formal result.

Proposition 3.3. *For any* $\mathbf{W} \subseteq \mathbf{U}$, $\Pi^{\mathbf{E}}_{\mathbf{X}}$ *with* $\mathbf{X} = \mathsf{PreCompl}_{\langle \mathbf{U}, \mathcal{F} \rangle} \langle W \rangle$ *is the least* Π*-view of* E *for which* $\mathcal{F}$ *embeds into* $\{\Pi^{\mathbf{E}}_{\mathbf{W}}, \Pi^{\mathbf{E}}_{\mathbf{X}}\}$. *It is thus a* Π*-precomplement of* $\Pi^{\mathbf{E}}_{\mathbf{W}}$. □

Example 3.4. Let $\mathbf{E}_1 = (R[\mathbf{U}_1], \mathcal{F}_1)$ with $\mathbf{U}_1 = ABCDEF$ and $\mathcal{F}_1 = \{A \to B, B \to C, CD \to E\})$. Let $\mathbf{W} = ABCE$, so the view whose precomplement is sought is $\Pi^{\mathbf{E}_1}_{\mathbf{W}} = \Pi^{\mathbf{E}_1}_{ABCE}$. Then $\mathsf{BaseCompl}_{\langle \mathbf{U}_1, \mathcal{F}_1 \rangle} \langle ABCE \rangle = ABCDEF \setminus ABCE = DF$ and $\mathsf{DPExt}_{\langle \mathbf{U}_1, \mathcal{F}_1 \rangle} \langle ABCE \rangle = CDE$, and so the least precomplement of $\Pi^{\mathbf{E}_1}_{ABCE}$ which preserves $\mathcal{F}'_1$ is $\Pi^{\mathbf{E}_1}_{CDE \cup DF} = \Pi^{\mathbf{E}_1}_{CDEF}$.

Discussion 3.5 (The inadequacy of canonical and minimal covers). Proposition 3.3 only identifies a precomplement which is optimal with respect to a fixed set $\mathcal{F}$ of FDs. In general, such a set of FDs will have many covers, with each giving rise to a possibly distinct precomplement. An optimal cover precomplement must be least when taken over all possible covers of $\mathcal{F}$. A potential strategy is to work with a cover of $\mathcal{F}$ which is normalized in some way which guarantees that the precomplement which it induces is no larger (in terms of $\preceq$) than any other.

There are a number of results which suggest that such an approach might be fruitful. In one of the earliest papers on the theory of FDs [5], it is shown that that any two nonredundant covers $\mathcal{C}_1$ and $\mathcal{C}_2$ of $\mathcal{F}$ have the remarkable property that for any FD $\varphi_1 \in \mathcal{C}_1$, there is an FD $\varphi_2 \in \mathcal{C}_2$ with the property that $\mathsf{LHS}(\varphi_1) \leftrightarrow \mathsf{LHS}(\varphi_2)$. Thus, up to FD equivalence, the left-hand sides of the FDs in any two nonredundant covers are the same. In [20], this result is strengthened: for any two minimum sets $\mathcal{C}_1$ and $\mathcal{C}_2$ of FDs, there is a bijective correspondence between the equivalent left-hand sides of each. In other words, the number of occurrences of an equivalent left-hand side in each set is the same.

Armed with these results, it would appear to be possible to work with minimum covers and then show that any two minimal precomplements differ only by an equivalence of attribute sets. Unfortunately, there are serious obstacles in the path of this potential tactic. The following examples illustrate the problems.

Examples 3.6 (The need for unique covers). Let $\mathbf{E}_2$ be the schema whose single relation symbol is $R[\mathbf{U}_2]$ with $\mathbf{U}_2 = ABCDE$, governed by the FDs in $\mathcal{F}_2 = \{A \to BCDE, CE \to D, CD \to E\}$. Then each of $\mathcal{F}_{21} = \{A \to BCD, CE \to D, CD \to E\}$ and $\mathcal{F}_{22} = \{A \to BCE, CE \to D, CD \to E\}$ is a minimum cover of $\mathcal{F}$. Rewriting these in RHS-simple format makes this clearer:

oarieli@mta.ac.il

$$\mathcal{F}'_{21} = \{A \to B, A \to C, A \to D, CE \to D, CD \to E\}$$
$$\mathcal{F}'_{22} = \{A \to B, A \to C, A \to E, CE \to D, CD \to E\}$$

Indeed, $A \to E$ may be derived from $A \to CD \to E$ and $A \to D$ may be derived from $A \to CE \to D$. Thus, either $A \to D$ or else $A \to E$, but not both, may be removed from $\mathcal{F}_2$ while retaining the same closure. From this observation and Proposition 3.3 it follows that $\Pi^{\mathbf{E}_2}_{BCDE}$ has two distinct minimal Π-precomplements, $\Pi^{\mathbf{E}_2}_{ABCD}$ and $\Pi^{\mathbf{E}_2}_{ABCE}$.

Note that the left-hand sides of the $\mathcal{F}_{2i}$'s are exactly the same in each case. Indeed, it is not difficult to see that $\mathsf{LHS}(\mathcal{C}_1) = \mathsf{LHS}(\mathcal{C}_2)$ for any two nonredundant minimum covers $\mathcal{C}_1$ and $\mathcal{C}_2$ of $\mathcal{F}_2$. It is the right-hand sides and only the right-hand sides which differ, and which lead to alternative minimal precomplements. Furthermore, $\mathcal{F}_2 \not\models D \leftrightarrow E$, so there can be no equivalence of the right-hand sides. Indeed, the two cover precomplements $\Pi^{\mathbf{E}_2}_{ABCD}$ and $\Pi^{\mathbf{E}_2}_{ABCE}$ are not equivalent in any simple sense.

This example does not rule out the possibility of identifying situations in which the right-hand sides of the FDs remain constant while the left-hand sides vary amongst equivalent sets. However, the following example suggests that this approach might be problematic in a more general setting.

Let $\mathbf{E}_3$ be the schema whose single relation symbol is $R[\mathbf{U}_3]$. with $\mathbf{U}_3 = A_1A_2B_1B_2CD$, governed by the FDs in $\mathcal{F}_3 = \{A_1B_1 \leftrightarrow A_2B_2, A_1B_1 \to C, A_2B_2 \to C, B_2 \to D\}$. A minimum cover is obtained by removing one of the FDs $A_1B_1 \to C$ or $A_2B_2 \to C$, but not both. Thus, the following two minimum covers are obtained.

$$\mathcal{F}'_{31} = \{A_1B_1 \leftrightarrow A_2B_2, A_1B_1 \to C, B_2 \to D\}$$
$$\mathcal{F}'_{32} = \{A_1B_1 \leftrightarrow A_2B_2, A_2B_2 \to C, B_2 \to D\}$$

Let $\mathbf{W}_3 = A_1B_1A_2B_2$, and consider the problem of finding an optimal precomplement of $\Pi^{\mathbf{E}}_{\mathbf{W}_3}$. Using the construction of Definition 3.2 and Proposition 3.3, the two precomplements $\Pi^{\mathbf{E}_3}_{\mathbf{W}_{31}}$ and $\Pi^{\mathbf{E}_3}_{\mathbf{W}_{32}}$, with respect to $\langle \mathbf{U}_3, \mathcal{F}'_{31} \rangle$ and $\langle \mathbf{U}_3, \mathcal{F}'_{32} \rangle$ respectively, have $\mathbf{W}_{31} = A_1B_1B_2CD$ and $\mathbf{W}_{32} = A_2B_2CD$. While these two sets are equivalent for $\mathcal{F}_3$, the relationship is not simple. The problem is that while $\mathcal{F} \models A_1B_1 \leftrightarrow A_2B_2$, A_1B_1 and A_2B_2 do not have the same substructure. If A_2B_2 is used, then B_2 may serve double duty as the left-hand side of $B_2 \to D$. On the other hand, if A_1B_1 is used, then B_2 must be included to cover $B_2 \to D$. Thus there is a sense in which A_1B_1 and A_2B_2 are not fully equivalent, and so it seems a bit of a stretch to regard $\Pi^{\mathbf{E}_3}_{\mathbf{W}_{31}}$ and $\Pi^{\mathbf{E}_3}_{\mathbf{W}_{32}}$ as similar enough to be the "same" precomplement under some form of isomorphism. For this reason, this sort of equivalence will not be allowed in the first instance. Rather, the approach is to first rule out allowing distinct precomplements which are equivalent via FD associations, and to reintroduce a limited form of equivalence subsequently as a special construction.

oarieli@mta.ac.il

Discussion 3.7 (Indecomposable FDs). Call the RHS-simple FD $\varphi \in \mathcal{F}^+$ *indecomposable* with respect to $\mathcal{F}$ if for every RHS-simple cover $\mathcal{C}$ of $\mathcal{F}$, $\varphi \in \mathcal{C}$. Put another way, if $\varphi = \mathbf{X} \to A$, then for every cover $\mathcal{C}$ of $\mathcal{F}$, there is an FD $\mathbf{X} \to \mathbf{Y} \in \mathcal{C}$ with $A \in \mathbf{Y}$. Define $\mathsf{Indecomp}\langle\mathcal{F}\rangle$ to be the set of all RHS-simple members of $\mathcal{F}^+$ which are indecomposable with respect to that set.

The following is immediate, invoking Proposition 3.3 for (b).

Proposition 3.8 (Indecomposable FDs and unique covers)

(a) *$\mathcal{F}$ admits a unique canonical cover iff $\mathsf{Indecomp}\langle\mathcal{F}\rangle$ is such a cover.*

(b) *If $\mathcal{F}$ admits a unique canonical cover $\mathcal{C}$, then for any $\mathbf{W} \subseteq \mathbf{U}$, $\Pi^{\mathbf{E}}_{\mathbf{W}'}$ is an optimal cover Π-precomplement of $\Pi^{\mathbf{E}}_{\mathbf{W}}$ for $\mathbf{W}' = \mathsf{PreCompl}_{\langle\mathbf{U},\mathcal{C}\rangle}\langle\mathbf{W}\rangle$.* □

Example 3.9 (Additional attributes necessary for meet complements). The example schema $\mathbf{E}_1$ of Definition 3.2 provides a setting with unique canonical covers. Indeed, $\mathcal{F}_1$ is its own unique canonical cover. Thus, in particular, $\Pi^{\mathbf{E}_1}_{CDEF}$ is the optimal cover Π-precomplement of $\Pi^{\mathbf{E}_1}_{ABCE}$.

However, $\Pi^{\mathbf{E}_1}_{CDEF}$ is not a complement of $\Pi^{\mathbf{E}_1}_{ABCE}$, since $ABCE \cap CDEF = CE$ is not a key for either $ABCE$ or $CDEF$ (see Summary 2.4). To remedy this situation, a set $\mathbf{X}$ of attributes must be added to $CDEF$ to render the attributes in $ABCE \cap (CDEF \cup \mathbf{X})$ a superkey for $ABCE$. The unique minimal solution is $\mathbf{X} = A$, so the optimal meet complement of $\Pi^{\mathbf{E}_1}_{ABCE}$ is $\Pi^{\mathbf{E}_1}_{ACDEF}$.

Definition 3.10 (Extension to optimal meet complements). To formalize this process, a few definitions are helpful. For these definitions, return to the general setting of Notation 3.1.

(a) $\mathsf{DPMut}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle = \mathbf{W} \cap \mathsf{DPExt}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$.

(b) $\mathsf{DPMutRHS}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle = \{A \in \mathbf{W} \mid \mathsf{DPMut}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle \to A \in \mathcal{F}^+\}$.

(c) $\mathsf{DPMut}\overline{\mathsf{RHS}}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle = \mathbf{W} \setminus \mathsf{DPMutRHS}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$.

Thus, $\mathsf{DPMut}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$ is the set of attribute which are mutual (i.e., common) to $\Pi^{\mathbf{E}_1}_{\mathbf{W}}$ and its optimal cover precomplement for $\mathcal{F}$. $\mathsf{DPMutRHS}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$ is the set of attributes in $\mathbf{W}$ which are already functionally determined by $\mathsf{DPMut}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$, and $\mathsf{DPMut}\overline{\mathsf{RHS}}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$ is the set of attributes in $\mathbf{W}$ which are not functionally determined by $\mathsf{DPMut}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$.

Example 3.11 (Continuation of Example 3.9). In Example 3.9, the definitions of Definition 3.10 yield $\mathsf{DPMut}_{\langle\mathbf{U}_1,\mathcal{F}_1\rangle}\langle ABCE\rangle = CE$, $\mathsf{DPMutRHS}_{\langle\mathbf{U}_1,\mathcal{F}_1\rangle}\langle ABCE\rangle = CE$, and $\mathsf{DPMut}\overline{\mathsf{RHS}}_{\langle\mathbf{U}_1,\mathcal{F}_1\rangle}\langle ABCE\rangle = ABCE \setminus CE = AB$. Only a subset of $\mathsf{DPMut}\overline{\mathsf{RHS}}_{\langle\mathbf{U}_1,\mathcal{F}_1\rangle}\langle ABCE\rangle$ which generates all of its attributes via FD closure which must be added to the mutual elements in order to obtain a meet complement. In this case, only only A need be added to the mutual elements, since $A \to B$. Hence the optimal meet complement of $\Pi^{\mathbf{E}_1}_{ABCE}$ is is $\Pi^{\mathbf{E}_1}_{A\cup CDEF} = \Pi^{\mathbf{E}_1}_{ACDEF}$. Unfortunately, such a unique minimal set of generating elements need not exist, even in the case that the canonical cover is unique, as shown by the example of the next paragraph.

oarieli@mta.ac.il

Example 3.12 (Non-uniqueness of extension from optimal cover precomplement to meet complement). Let $\mathbf{E}_4 = (R[\mathbf{U}_4], \mathcal{F}_4)$ with $\mathbf{U}_4 = ABCD$ and $\mathcal{F}_4 = \{AB \to C, C \to A, D \to A\}$. Then $\mathcal{F}_4$ is its own unique canonical cover, and so the construction of Proposition 3.3 delivers a unique cover precomplement for each subset of $\mathbf{U}$. However, it is not necessarily the case that such a precomplement has a unique extension to a meet complement. The optimal cover precomplement of $\Pi^{\mathbf{E}_4}_{ABC}$ is $\Pi^{\mathbf{E}_4}_{AD}$, but $\mathsf{DPMut}_{\langle \mathbf{U}_4, \mathcal{F}_4 \rangle}\langle ABC \rangle = A$ is not a key for either ABC or BD. Here $\mathsf{DPMutRHS}_{\langle \mathbf{U}_4, \mathcal{F}_4 \rangle}\langle ABC \rangle = A$ and $\mathsf{DPMut}\overline{\mathsf{RHS}}_{\langle \mathbf{U}_4, \mathcal{F}_4 \rangle}\langle ABC \rangle = BC$, but only one of B or C need be added to $\mathsf{DPMut}_{\langle \mathbf{U}_4, \mathcal{F}_4 \rangle}\langle ABC \rangle$ to make it a key for $\Pi^{\mathbf{E}_1}_{ABC}$. Thus each of $\Pi^{\mathbf{E}_4}_{ABD}$ and $\Pi^{\mathbf{E}_4}_{ACD}$ is a fully minimal Π-complement of $\Pi^{\mathbf{E}_4}_{ABC}$, so it has no optimal meet Π-complement. Hence, requiring unique canonical covers is not sufficient to guarantee universal (i.e., fully optimal) Π-complements. The simplest remedy is to require that the set of FDs admit a unique key, as developed below.

In [6, Cor. 3], it is shown that a general dependency-preserving decomposition of a universal schema $\mathbf{E}$ (into possibly more than just two Π-views) is lossless if and only if the attributes of at least one of those views forms a superkey for $\mathbf{E}$. For a decomposition into two components, this result can be strengthened and related to the classical result [25, Thm. 2] as follows.

Lemma 3.13 (Meet complements and superkeys). *Let $\{\Pi^{\mathbf{E}}_{\mathbf{W}_1}, \Pi^{\mathbf{E}}_{\mathbf{W}_2}\}$ be a covered pair of views of $\mathbf{E}$. Then for $i \in \{1, 2\}$, $\mathbf{W}_1 \cap \mathbf{W}_2 \to \mathbf{W}_i \in \mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle)^+$ iff $\mathbf{W}_{3-i}$ is a superkey; i.e., $\mathbf{W}_{3-i} \to \mathbf{U} \in \mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle)^+$.*

Proof. Let $i \in \{1, 2\}$. If $\mathbf{W}_1 \cap \mathbf{W}_2 \to \mathbf{W}_i \in \mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle)^+$, then it is immediate that $\mathbf{W}_{3-i}$ is a superkey. Conversely, suppose that $\mathbf{W}_{3-i}$ is a superkey. Let $M_1, M_2 \in \mathsf{LDB}(\mathbf{E})$ have the property that they agree on the attributes of $\mathbf{W}_1 \cap \mathbf{W}_2$; i.e., $\pi_{\mathbf{W}_1 \cap \mathbf{W}_2}(M_1) = \pi_{\mathbf{W}_1 \cap \mathbf{W}_2}(M_2)$. Let M be any relation on $\mathbf{U}$ with $\pi_{\mathbf{W}_i}(M) = \pi_{\mathbf{W}_i}(M_i)$ and $\pi_{\mathbf{W}_{3-i}}(M) = \pi_{\mathbf{W}_{3-i}}(M_{3-i})$. Such an M always exists; indeed, the natural join $\pi^{\mathbf{E}}_{\mathbf{W}_1}(M_i) \bowtie \pi^{\mathbf{E}}_{\mathbf{W}_2}(M_{3-i})$ is one such relation. Since such a relation satisfies all FDs in $\mathcal{F}$ by construction, $M \in \mathsf{LDB}(\mathbf{E})$. Since $\mathbf{W}_{3-i}$ is a superkey, $M = M_{3-i}$, whence $\pi_{\mathbf{W}_i}(M_{3-i}) = \pi_{\mathbf{W}_i}(M)$, and since $\pi_{\mathbf{W}_i}(M) = \pi_{\mathbf{W}_i}(M_i)$ by construction, it follows that $\pi_{\mathbf{W}_i}(M_{3-i}) = \pi_{\mathbf{W}_i}(M_i)$; i.e., $\pi_{\mathbf{W}_i}(M_1) = \pi_{\mathbf{W}_i}(M_2)$. Since the only initial assumption about M_1 and M_2 is that they agree on $\mathbf{W}_1 \cap \mathbf{W}_2$, it follows that $\mathbf{W}_1 \cap \mathbf{W}_2 \to \mathbf{W}_i \in \mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle)^+$, as required. □

Definition 3.14 (The strong cover property). Say that $\langle \mathbf{U}, \mathcal{F} \rangle$ has the *strong cover property* if $\mathcal{F}$ admits a unique canonical cover and $\langle \mathbf{U}, \mathcal{F} \rangle$ has just one key. In view of the discussion in Summary 2.2, this latter condition may be stated equivalently as requiring that $\mathcal{F}$ admit a unique key. In other words, $\langle \mathbf{U}, \mathcal{F} \rangle$ has the strong cover property if $\mathcal{F}$ admits both a unique canonical cover and a unique key, so this property does not depend upon attributes in $\mathbf{U}$ which do not occur in $\mathcal{F}$. However, the key itself for $\langle \mathsf{Attr}_{\cup}(\mathcal{F}), \mathcal{F} \rangle$ will in fact differ from that of $\langle \mathbf{U}, \mathcal{F} \rangle$ if $\mathbf{U} \neq \mathsf{Attr}_{\cup}(\mathcal{F})$. In the case that $\langle \mathbf{U}, \mathcal{F} \rangle$ admits a unique key, $\mathsf{Key}_{\langle \mathbf{U}, \mathcal{F} \rangle}$ will be used to denote that key. For further studies of schemata which admit unique keys, see [7] and [2, Thm. 3.4].

oarieli@mta.ac.il

The main result on optimal meet Π-complements may now be presented. Define $\mathsf{MeetCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle$ to be:

(mc-i) $\mathsf{PreCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle$ if $\mathsf{DPMut}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle \rightarrow \mathsf{PreCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle \in \mathsf{FDAug}(\langle \mathbf{U},\mathcal{F}\rangle)^+$

(mc-ii) $\mathsf{PreCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle \cup \mathsf{Key}_{\langle \mathbf{U},\mathcal{F}\rangle}$ otherwise

Theorem 3.15 (Universal Π-complements). *If $\langle \mathbf{U},\mathcal{F}\rangle$ has the strong cover property, then for any $\mathbf{W} \subseteq \mathbf{U}$, $\Pi^{\mathbf{E}}_{\mathbf{W}}$ admits a universal Π-complement $\Pi^{\mathbf{E}}_{\mathbf{X}}$, with $\mathbf{X} = \mathsf{MeetCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle$. In words, the universal Π-complement of $\Pi^{\mathbf{E}}_{\mathbf{W}}$ is just the optimal cover Π-precomplement if $\mathsf{DPMut}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle$ is a superkey for $\mathsf{PreCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle$, and it is the projection defined by augmenting the attributes of $\mathsf{PreCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle$ with the unique key $\mathsf{Key}_{\langle \mathbf{U},\mathcal{F}\rangle}$ otherwise.*

Proof. If $\mathsf{DPMut}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle \rightarrow \mathsf{PreCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle \in \mathsf{FDAug}(\langle \mathbf{U},\mathcal{F}\rangle)^+$, the optimal meet Π-precomplement is already a meet complement (see Summary 2.4). Otherwise, for $\Pi^{\mathbf{E}}_{\mathbf{X}}$ to be a meet complement of $\Pi^{\mathbf{E}}_{\mathbf{W}}$, it is necessary that $\mathbf{W} \cap \mathbf{X}$ be a key for $\mathbf{W}$. In view of Lemma 3.13, this holds iff $\mathbf{X}$ is a superkey for $\langle \mathbf{U},\mathcal{F}\rangle$. To make $\mathbf{X}$ such a superkey, the least set attributes which must be added to $\mathsf{PreCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle$ is precisely those elements of $\mathsf{Key}_{\langle \mathbf{U},\mathcal{F}\rangle}$ which are not already in $\mathsf{PreCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle$; exactly as (mc-ii) dictates.

Note that if $\mathsf{Key}_{\langle \mathbf{U},\mathcal{F}\rangle} \subseteq \mathsf{PreCompl}_{\langle \mathbf{U},\mathcal{F}\rangle}\langle \mathbf{W}\rangle$ already, then adding $\mathsf{Key}_{\langle \mathbf{U},\mathcal{F}\rangle}$ has no effect, but specifying that it be added in all cases does no harm. □

4 Quasi-Universal Π-Complements

The conditions imposed in Theorem 3.15 in order to obtain optimal meet complements for all Π-views of a given schema are quite strong. In particular, they rule out most situations with multiple (candidate) keys. Since such multiple keys are quite common in practice, it is important to seek extensions which relax, at least to some degree, the strict uniqueness properties imposed by the strong cover property. In this section, one such extension is pursued, in which individual attributes may be equivalent to one another. An example will help to identify the main ideas.

Example 4.1. Let $\mathbf{E}_5 = (R[\mathbf{U}_5], \mathcal{F}_5)$ with $\mathbf{U}_5 = AB_1B_2B_3C$ and $\mathcal{F}_5 = \{B_1 \leftrightarrow B_2, B_2 \leftrightarrow B_3, B_1 \rightarrow C, B_2 \rightarrow C, B_3 \rightarrow C\}$. Since $\mathcal{F}_5$ contains (simple) equivalences, it admits neither a a unique canonical cover nor a unique key. Indeed, each of

$$\mathcal{F}'_{51} = \{B_1 \rightarrow B_2, B_2 \rightarrow B_3, B_3 \rightarrow B_1, B_1 \rightarrow C\}$$
$$\mathcal{F}'_{52} = \{B_1 \rightarrow B_2, B_2 \rightarrow B_3, B_3 \rightarrow B_1, B_2 \rightarrow C\}$$
$$\mathcal{F}'_{53} = \{B_1 \rightarrow B_2, B_2 \rightarrow B_3, B_3 \rightarrow B_1, B_3 \rightarrow C\}$$

is a minimum and canonical cover, and the view $\Pi^{\mathbf{E}_5}_{AB_1B_2B_3}$ has three distinct minimal meet Π-complements, $\Pi^{\mathbf{E}_5}_{B_1C}$, $\Pi^{\mathbf{E}_5}_{B_2C}$, and $\Pi^{\mathbf{E}_5}_{B_3C}$. Furthermore, each of

oarieli@mta.ac.il

AB_1, AB_2, and AB_3 is a key for $\langle \mathbf{U}_5, \mathcal{F}_5 \rangle$. Nevertheless, these three are closely related in the sense that each one is obtained from the other by replacing each attribute by an equivalent one. This is also the case for the three distinct canonical covers above, as well as the three distinct keys. Indeed, if the three attributes in $B_1 B_2 B_3$ are collapsed into one, say B_1, with the resulting schema $\widetilde{\mathbf{E}}_5 = (R[\widetilde{\mathbf{U}}_5], \widetilde{\mathcal{F}}_5)$ with $\widetilde{\mathcal{F}}_5 = \{B_1 \to C\}$, then $\langle \widetilde{\mathbf{U}}_5, \widetilde{\mathcal{F}}_5 \rangle$ has the strong cover property and the universal Π-complement of $\Pi^{\widetilde{\mathbf{E}}_5}_{AB_1}$ is $\Pi^{\widetilde{\mathbf{E}}_5}_{B_1 C}$.

The fully minimal Π-complements may be recovered easily from this solution. For example, if the view to be complemented is $\Pi^{\mathbf{E}_5}_{AB_1 B_2 B_3}$, then first compute the complement of $\Pi^{\widetilde{\mathbf{E}}_5}_{AB_1}$, which is $\Pi^{\widetilde{\mathbf{E}}_5}_{B_1 C}$. Then, restore the individual values associated with B_1, placing as few as possible in the complement. In general, there will be several ways to do this. In the example, one of $B_1 B_2 B_3$ is placed in the complement, with the other two in the main view. This leads exactly to the three complements identified above.

A formalization of these ideas makes up the rest of this section.

Definition 4.2 (Simple equivalence of attributes and quasi-universal Π-complements). Recall that two attributes A and B are *simply equivalent* in $\mathcal{F}$ if $\mathcal{F} \models A \leftrightarrow B$. For convenience, let $\mathsf{SimpEquiv}(\mathcal{F})$ denote the set of all simple equivalences implied by $\mathcal{F}$; thus $\mathsf{SimpEquiv}(\mathcal{F}) = \{A \leftrightarrow B \mid \mathcal{F} \models A \leftrightarrow B\}$. Define $\equiv^1_{\langle \mathbf{U}, \mathcal{F} \rangle}$ to be the equivalence relation on $\mathbf{U}$ given by $A \equiv^1_{\langle \mathbf{U}, \mathcal{F} \rangle} B$ iff $\mathsf{FDAug}(\langle \mathbf{U}, \mathcal{F} \rangle) \models A \leftrightarrow B$.

Given $\mathbf{W}, \mathbf{W}' \subseteq \mathbf{U}$, $\Pi^{\mathbf{E}}_{\mathbf{W}'}$ is called a *quasi-universal Π-complement* of $\Pi^{\mathbf{E}}_{\mathbf{W}}$ with respect to $\equiv^1_{\langle \mathbf{U}, \mathcal{F} \rangle}$ if for any other minimal meet Π-complement $\Pi^{\mathbf{E}}_{\mathbf{W}''}$, there is a bijection $\beta : \mathbf{W}' \to \mathbf{W}''$ with the property that for each $A \in \mathbf{W}'$, $A \equiv^1_{\langle \mathbf{U}, \mathcal{F} \rangle} \beta(A)$. In other words, a minimal meet complement is quasi-universal with respect to $\equiv^1_{\langle \mathbf{U}, \mathcal{F} \rangle}$ if all other meet complements differ only by simple equivalences on the attributes. A universal complement is a special case in which the only such bijection β is the identity on $\mathbf{W}'$. Since all quasi-universal complements in this paper will be with respect to the simple equivalence of the underlying schema, that qualification will be dropped, and it will simply be said that one view is a *quasi-universal complement* of another.

Definition 4.3 (Aliases and representation functions). Any $B \in \mathbf{U}$ with $A \equiv^1_{\langle \mathbf{U}, \mathcal{F} \rangle} B$ is called an *alias* for A in $\langle \mathbf{U}, \mathcal{F} \rangle$. Call A *aliased* in $\langle \mathbf{U}, \mathcal{F} \rangle$ if it has an alias other than itself. Otherwise, call A *unaliased* in $\langle \mathbf{U}, \mathcal{F} \rangle$. The set of all aliases of A in $\langle \mathbf{U}, \mathcal{F} \rangle$ is denoted $\mathsf{Alias}_{\langle \mathbf{U}, \mathcal{F} \rangle}(A)$. $\widetilde{\mathbf{U}} = \{\mathsf{Alias}_{\langle \mathbf{U}, \mathcal{F} \rangle}(A) \mid A \in \mathbf{U}\}$; thus, $\widetilde{\mathbf{U}}$ is just the set of all equivalence classes of $\equiv^1_{\langle \mathbf{U}, \mathcal{F} \rangle}$. For $\mathbf{W} \subseteq \mathbf{U}$, define $\widetilde{\mathbf{W}} = \{\mathsf{Alias}_{\langle \mathbf{U}, \mathcal{F} \rangle}(A) \mid A \in \mathbf{W}\}$. It is useful to have a more compact representation for elements of the form $\mathsf{Alias}_{\langle \mathbf{U}, \mathcal{F} \rangle}(A)$. To this end, define a *representation* for $\widetilde{\mathbf{U}}$ to be a function $\mathfrak{r} : \mathbf{U} \to \widetilde{\mathbf{U}}$ with the properties that for each $A \in \mathbf{U}$, $\mathfrak{r}(A) \in \mathsf{Alias}_{\langle \mathbf{U}, \mathcal{F} \rangle}(A)$ and $\mathfrak{r}(A) = \mathfrak{r}(B)$ iff $B \in \mathsf{Alias}_{\langle \mathbf{U}, \mathcal{F} \rangle}(A)$. Thus, a representation maps all elements of a given class $\mathsf{Alias}_{\langle \mathbf{U}, \mathcal{F} \rangle}(A)$ to the same element, which must itself be a member of $\mathsf{Alias}_{\langle \mathbf{U}, \mathcal{F} \rangle}(A)$. In the above example, the representation used is $\mathfrak{r}_5$ defined by $\mathfrak{r}_5(A) = A$, $\mathfrak{r}_5(C) = C$, and $\mathfrak{r}_5(B_1) = \mathfrak{r}_5(B_2) = \mathfrak{r}_5(B_3) = B_1$.

oarieli@mta.ac.il

Fix a representation function $\mathfrak{r}$ for $\mathcal{F}$, and define $\widetilde{\mathcal{F}}$ to be the set of FDs obtained from $\mathcal{F}$ by replacing each $A \in \mathsf{Attr}(\mathcal{F})$ with $\mathfrak{r}(A)$, and then removing all trivial FDs. In Example 4.1, $\widetilde{\mathcal{F}}_5$ is obtained in this way from $\mathcal{F}_5$. Choosing a different representation function will change the names of the symbols and nothing else, so it is justified to write $\mathcal{F}$ without any reference to $\mathfrak{r}$.

Finally, define $\widetilde{\mathbf{E}} = (R[\widetilde{\mathbf{U}}], \widetilde{\mathcal{F}})$, the *simple-alias reduction* of $\mathbf{E}$.

It is perhaps worth noting that a representation as defined above is a special case of a *system of distinct representatives (SDR)*, also called a *transversal* [22], [8, p. 192], [23, Def. 1]. In a general SDR, the sets to be represented may overlap, and so the question of existence is nontrivial; the solution being given by *Hall's Marriage Theorem* (see the references identified above). In the special case considered here, the sets do not overlap and so the existence of an SDR is trivial.

Example 4.4 (An issue surrounding simple-alias reduction). The idea is to represent the original $\langle \mathbf{U}, \mathcal{F} \rangle$ using $\widetilde{\mathcal{F}}$ together with a set $\mathcal{E}$ of simple FDs which defines all of the simple equivalences of $\mathcal{F}$. Then, the different minimal complements may be obtained by using the "unique" canonical cover $\widetilde{\mathcal{F}}$ as the representation function $\mathfrak{r}$ varies. This almost works; however, there is one minor complication, which is best illustrated by example. Let $\mathbf{E}_6 = (R[A_1A_2BC], \mathcal{F}_6)$ be the relational schema with $\mathcal{F}_6 = \{A_1 \leftrightarrow A_2, A_2 \leftrightarrow BC\}$. Then, with $\mathfrak{r}_6(A_1) = \mathfrak{r}_6(A_2) = A_2$, $\widetilde{\mathcal{F}}_6 = \{A_2 \leftrightarrow BC\}$ which is its own unique canonical cover. The view $\Pi^{\mathbf{E}_6}_{A_2BC}$ has two minimal Π-complements, $\Pi^{\mathbf{E}_6}_{A_1A_2}$ and $\Pi^{\mathbf{E}_6}_{A_1BC}$ because, in addition to the alias A_2, BC is also formally equivalent to A_1. This property depends upon the isolation of BC from other FDs. For example, let $\mathbf{E}'_6 = (R[A_1A_2BCD], \mathcal{F}'_6)$ be the relational schema with $\mathcal{F}'_6 = \mathcal{F}_6 \cup \{BC \rightarrow D\}$. With this simple addition, $\widetilde{\mathcal{F}}'_6 = \{A_2 \leftrightarrow BC, BC \rightarrow D\}$ no longer has the strong cover property. Nevertheless, the case illustrated by $\mathbf{E}_6$ must be ruled out.

Definition 4.5 (The quasi-strong cover property). For $\mathbf{X} \subseteq \mathbf{U}$ with $\mathsf{Card}(\mathbf{X}) > 1$ (i.e., $\mathbf{X}$ contains at least two distinct elements) and $A, B \in \mathbf{U}$, call $\{A, B, \mathbf{X}\}$ a *complex triple* for $\mathcal{F}$ if both $A \leftrightarrow B, B \leftrightarrow \mathbf{X} \in F^+$, and for no proper subset $\mathbf{X}' \subsetneq \mathbf{X}$ is it the case that $\mathbf{X}' \leftrightarrow B \in \mathcal{F}^+$. In other words, $A \leftrightarrow B \leftrightarrow \mathbf{X}$ holds with $\mathbf{X}$ consisting of at least two attributes, and no element of $\mathbf{X}$ may be removed while retaining that property. For both $\mathcal{F}_6$ and $\mathcal{F}'_6$, $\{A_1, A_2, BC\}$ is a complex triple. Say that $\mathcal{F}$ is *free of complex triples* if $\mathcal{F}^+$ contains no such triples. Define $\mathcal{F}$ to have the *quasi-strong cover property* if it is free of complex triples and has the strong equivalence-cover property. It is worth emphasizing that requiring that $\mathcal{F}$ be free of complex triples does not impose substantial further limits beyond those already implied by the strong cover property on $\widetilde{\mathcal{F}}$, since as illustrated by the schema $\mathbf{E}'_6$, if the attributes of a complex triple are connected to other attributes via additional FDs, then the strong cover property is lost anyway.

Lemma 4.6 (Derivation of simple equivalences). *If $\mathcal{F}$ is free of complex triples and $\mathcal{C}$ is a canonical cover for $\mathcal{F}$, then for every simple equivalence*

oarieli@mta.ac.il

$A \leftrightarrow B \in \mathcal{F}^+$ with $A \neq B$, there are $A_1, \ldots, A_k \in \mathbf{U}$ with $A_1 = A$, $A_k = B$, and $A_i \rightarrow A_{i+1} \in \mathcal{C}$ for $i \in 1, \ldots, k-1$.

Proof outline. Without loss of generality, assume that $\mathcal{F}$ is canonical. The proof is based upon the claim that there must be a sequence $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_k$ of subsets of $\mathbf{U}$ with $\mathbf{X}_1 = A$, $\mathbf{X}_k = B$, and for each $i \in \{1, \ldots, k-1\}$, $\mathbf{X}_i \rightarrow C \in \mathcal{F}$ for each $C \in \mathbf{X}_{i+1}$. Then, using the property that $\mathcal{F}$ be free of complex triples, it follows that each $\mathbf{X}_i$ must consist of single attribute, whence the result.

To place this argument on rigorous grounds, a deductive formalism for FDs is required. The most appropriate one in this case is the *derivation directed acyclic graphs* or *DDAGs* [20], [21, Sec. 4.5]. In this case, there must be a DDAG with single initial vertex labelled with A and a final vertex labelled B. Furthermore, there must be such a graph with exactly one outgoing edge from each vertex other than that labelled with B; otherwise, it is possible to deduce a complex cycle. The full elaboration is straightforward but tedious; space limitation preclude expanding it further. □

Proposition 4.7 (Representation of quasi-strong covers). *If $\mathcal{F}$ has the quasi-strong cover property, then every canonical cover of $\mathcal{F}$ is of the form $\mathcal{C}_{\mathsf{Str}} \cup \mathcal{C}_{\mathsf{Eq}}$, with the following properties.*

(a) *$\mathcal{C}_{\mathsf{Str}}$ is the unique canonical cover for $\langle \widetilde{\mathbf{U}}, \widetilde{\mathcal{F}} \rangle$ for some representation function.*

(b) *$\mathcal{C}_{\mathsf{Eq}}$ is a canonical cover for $\mathsf{SimpEquiv}(\mathcal{F})$.*

Proof. In view of Lemma 4.6, every canonical cover of $\mathcal{F}$ must contain a subset consisting of simple equivalences which form a cover for $\mathsf{SimpEquiv}(\mathcal{F})$. Let $\mathcal{C}_{\mathsf{Eq}}$ be any such set. Since $\mathcal{F}$ is assumed to have the quasi-strong cover property, the remainder of the canonical cover must be $\mathcal{F}_{\mathsf{Str}}$. □

Definition 4.8 (Constructions for quasi-universal Π-complements). The precise construction of the quasi-universal Π-complements of $\Pi^{\mathbf{E}}_{\mathbf{W}}$ may now be specified. Although the definitions are straightforward, it may prove useful to follow them in parallel with Example 4.10, which illustrates their application to a fixed schema.

Let $\mathbf{W} \subseteq \mathbf{U}$. The process of working with $\langle \widetilde{\mathbf{U}}, \widetilde{\mathcal{F}} \rangle$ rather than $\langle \mathbf{U}, \mathcal{F} \rangle$ collapses all simply equivalent elements into one. In order to recover the views corresponding to $\langle \mathbf{U}, \mathcal{F} \rangle$, it is necessary to "uncollapse" these equivalent elements in the appropriate fashion. To begin, the attributes in $\mathbf{W}$ which have an alias in the base complement of $\mathbf{W}$ are identified.

(a) $\mathsf{DPBoth}_{\langle \mathbf{U}, \mathcal{F} \rangle}\langle \mathbf{W} \rangle = \{A \in \mathbf{W} \mid \mathsf{BaseCompl}_{\langle \mathbf{U}, \mathcal{F} \rangle}\langle \mathbf{W} \rangle \cap \mathsf{Alias}_{\langle \mathbf{U}, \mathcal{F} \rangle}(A) \neq \emptyset\}$

In order to maintain an embedded cover of $\mathcal{C}_{\mathsf{Eq}}$ as identified in Proposition 4.7 above, if an attribute A_1 occurs in the view $\Pi^{\mathbf{E}}_{\mathbf{W}}$ to be complemented while an equivalent attribute A_2 occurs in a potential complement $\Pi^{\mathbf{E}}_{\mathbf{X}}$ but not in $\Pi^{\mathbf{E}}_{\mathbf{W}}$, then either one of these, or else a third attribute equivalent to both, must appear in $\mathbf{W} \cap \mathbf{X}$. This alias must be an element of $\mathbf{W}$, since the meet is a subset of $\mathbf{W}$. Additionally, any equivalence class which has a representative in

oarieli@mta.ac.il

$\mathsf{DPMut}_{\langle\widetilde{\mathbf{U}},\widetilde{\mathcal{F}}\rangle}\langle\widetilde{\mathbf{W}}\rangle$ must also have at least one representative in the meet of $\Pi^{\mathbf{E}}_{\mathbf{W}}$ and any of its minimal complements. The possible elements for the meet are defined as follows.

(b) $\mathsf{DPMeetSets}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle = \{\mathbf{W} \cap \mathsf{Alias}_{\langle\mathbf{U},\mathcal{F}\rangle}(A) \mid (A \in \mathsf{DPBoth}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle) \text{ or } (\mathfrak{r}(A) \in \mathsf{DPMut}_{\langle\widetilde{\mathbf{U}},\widetilde{\mathcal{F}}\rangle}\langle\widetilde{\mathbf{W}}\rangle)\}$

(c) $\mathsf{DPMeetAlts}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$ is all subsets of $\mathbf{W}$ formed by selecting exactly one attribute from each member of $\mathsf{DPMeetSets}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$.

Some set of attributes in $\mathsf{DPMeetAlts}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$ must be contained in the meet of any quasi-universal Π-complement. Observe in particular that if an attribute has no alias other than itself, then it will be in the meet of $\Pi^{\mathbf{E}}_{\mathbf{W}}$ and its minimal complement iff it occurs in the meet of $\Pi^{\widetilde{\mathbf{E}}}_{\widetilde{\mathbf{W}}}$ and its optimal complement.

Finally, the alternatives for the quasi-universal Π-complements may be specified by adding to the base complement a set of suitable elements for the meet.

(d) $\mathsf{MeetComplAlts}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle = \{\mathbf{X} \cup \mathsf{BaseCompl}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle \mid \mathbf{X} \in \mathsf{DPMeetAlts}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle\}$

This is summarized in the following theorem. The proof is a straightforward argument based upon the above constructions, and is omitted due to space considerations.

Theorem 4.9. *Let* $\mathbf{W} \subseteq \mathbf{U}$*, and assume that* $\langle\mathbf{U},\mathcal{F}\rangle$ *has the quasi-strong cover property. Then* $\Pi^{\mathbf{E}}_{\mathbf{X}}$ *is a quasi-universal* Π*-complement of* $\Pi^{\mathbf{E}}_{\mathbf{W}}$ *iff* $\mathbf{X} \in \mathsf{MeetComplAlts}_{\langle\mathbf{U},\mathcal{F}\rangle}\langle\mathbf{W}\rangle$. □

Example 4.10. Define the schema

$$\mathbf{E}_7 = (R[A_1A_2A_3BC_1C_2C_3DE_1E_2E_3F_1F_2F_3GHIJ_1J_2J_3K_1K_2K_3], \mathcal{F}_7)$$

with $\mathcal{F}_7 = \mathcal{C}_{\mathsf{Str7}} \cup \mathcal{C}_{\mathsf{Eq7}}$ and these latter two sets given by

$$\begin{aligned}\mathcal{C}_{\mathsf{Str7}} &= \{A_1B \to C_1D,\ C_1D \to E_1F_1G,\ E_1F_1 \to H,\ H \to IJ_1K_1\}\\ \mathcal{C}_{\mathsf{Eq7}} &= \{A_1 \to A_2,\ A_2 \to A_3,\ A_3 \to A_1, C_1 \to C_2,\ C_2 \to C_3,\ C_3 \to C_1,\\ &\quad E_1 \to E_2,\ E_2 \to E_3,\ E_3 \to E_1, F_1 \to F_2,\ F_2 \to F_3,\ F_3 \to F_1,\\ &\quad J_1 \to J_2,\ J_2 \to J_3,\ J_3 \to J_1, K_1 \to K_2,\ K_2 \to K_3,\ K_3 \to K_1\}\end{aligned}$$

Upon RHS-simplification, $\mathcal{C}_{\mathsf{Str7}}$ becomes a unique canonical cover for $\widetilde{\mathcal{F}}_7$, while $\mathcal{C}_{\mathsf{Eq7}}$ is a canonical cover for $\mathsf{SimpEquiv}(\mathcal{F})$, with $\alpha_i \equiv^1_{\langle\mathbf{U}_7,\mathcal{F}_7\rangle} \alpha_j$ for $\alpha \in \{A, C, E, F, J, K\}$ and $i \in \{1, 2, 3\}$. The representation used is $\mathfrak{r}_7 : \alpha_i \mapsto \alpha_1$ for $\alpha \in \{A, C, E, F, J, K\}$ and $\mathfrak{r}_7 : \alpha \mapsto \alpha$ for $\alpha \in \{B, D, G, H, I\}$.

Let $\mathbf{W} = E_1E_2E_3F_1F_2GHIJ_1K_1K_2K_3$ with $\Pi^{\mathbf{E}_7}_{\mathbf{W}}$ the view to be complemented. First of all, for $\widetilde{\mathbf{E}}_7 = (R[A_1BC_1DE_1F_1GHIJ_1K_1], \mathcal{C}_{\mathsf{Str7}})$, using the constructions for the strong cover property, $\widetilde{\mathbf{W}} = E_1F_1GHIJ_1K_1$, $\mathsf{BaseCompl}_{\langle\widetilde{\mathbf{U}},\widetilde{\mathcal{F}}\rangle}\langle\widetilde{\mathbf{W}}\rangle = A_1BC_1, D$, $\mathsf{DPExt}_{\langle\widetilde{\mathbf{U}},\widetilde{\mathcal{F}}\rangle}\langle\widetilde{\mathbf{W}}\rangle = A_1BC_1DE_1F_1G$, and $\mathsf{DPMut}_{\langle\widetilde{\mathbf{U}},\widetilde{\mathcal{F}}\rangle}\langle\widetilde{\mathbf{W}}\rangle = E_1F_1G$. It is easy to see that A_1B is a key for $\widetilde{\mathbf{E}}_7$, so by

oarieli@mta.ac.il

Theorem 3.15 the optimal complement of $\Pi^{\widetilde{\mathbf{E}}_7}_{\widetilde{\mathbf{W}}}$ is $\Pi^{\widetilde{\mathbf{E}}_7}_{\mathbf{X}}$ with $\mathbf{X} = \mathsf{PreCompl}_{\langle \widetilde{\mathbf{U}}_7, \widetilde{\mathcal{F}}_7 \rangle}\langle \mathbf{W} \rangle = \mathsf{BaseCompl}_{\langle \widetilde{\mathbf{U}}_7, \widetilde{\mathcal{F}}_7 \rangle}\langle \mathbf{W} \rangle \cup \mathsf{DPExt}_{\langle \mathbf{U}_7, \mathcal{F}_7 \rangle}\langle \mathbf{W} \rangle = A_1BC_1DE_1F_1G$. The task is to uncollapse this solution into one for $\mathbf{E}_7$ and the view $\Pi^{\mathbf{E}_7}_{\mathbf{W}}$.

First of all, $\mathsf{DPBoth}_{\langle \mathbf{U}_7, \mathcal{F}_7 \rangle}\langle \mathbf{W} \rangle = F_1F_2J_1$, the elements of $\mathbf{W}$ which have an alias in $\mathsf{BaseCompl}_{\langle \mathbf{U}_7, \mathcal{F}_7 \rangle}\langle \mathbf{W} \rangle = A_1A_2A_3BC_1C_2C_3DF_3J_2J_3$. Next, $\mathsf{DPMeetSets}_{\langle \mathbf{U}_7, \mathcal{F}_7 \rangle}\langle \mathbf{W} \rangle = \mathbf{W} \cap (F_1F_2F_3J_1J_2J_3 \cup E_1E_2E_3F_1F_2F_3G) = E_1E_2E_3F_1F_2GJ_1$ consists of all attributes in $\mathbf{W}$ with an alias in either $\mathsf{DPBoth}_{\langle \mathbf{U}_7, \mathcal{F}_7 \rangle}\langle \mathbf{W} \rangle$ or $\mathsf{DPMut}_{\langle \widetilde{\mathbf{U}}, \widetilde{\mathcal{F}} \rangle}\langle \widetilde{\mathbf{W}} \rangle$. These are the possible elements for inclusion in the meet. The actual meet selects one element from $\mathsf{DPMeetSets}_{\langle \mathbf{U}_7, \mathcal{F}_7 \rangle}\langle \mathbf{W} \rangle$ for each equivalence class of $\equiv^1_{\langle \mathbf{U}_7, \mathcal{F}_7 \rangle}$ which is represented in that set. Thus the set of meet alternatives is given by $\mathsf{DPMeetAlts}_{\langle \mathbf{U}_7, \mathcal{F}_7 \rangle}\langle \mathbf{W} \rangle = \{E_iF_jGJ_1 \mid i \in \{1,2,3\} \text{ and } j \in \{1,2\}\}$, and so $\mathsf{MeetComplAlts}_{\langle \mathbf{U}_7, \mathcal{F}_7 \rangle}\langle \mathbf{W} \rangle = A_1A_2A_3BC_1C_2C_3DF_3GJ_1J_2J_3 \cup \{E_iF_j \mid i \in \{1,2,3\} \text{ and } j \in \{1,2\}\}$. Hence, there are six alternatives for a quasi-universal Π-complement, all related by the replacement of simple attributes by equivalent ones. Note, however, that not any equivalent attribute will do. F_1 may only be replaced with F_2, not F_3, and J_1 may not be replaced by either J_2 or J_3.

5 Conclusions and Further Directions

A characterization of universal complements in the setting of projections of a universal relational schema which is constrained by functional dependencies has been presented. Although the conditions are rather strict — existence of a unique canonical cover and a unique key — they reveal the nature of requirements which must be imposed in order to realize such complements. Conditions which allow a modest amount of flexibility — by replacing single attributes by equivalent ones to yield quasi-universal complements, have also been developed. They give useful solutions in certain important settings, such as when there are multiple candidate keys.

There are several key areas for further work on this subject.

Extension of the conditions: The presentations of Sections 3 and 4 were designed to illustrate the fundamental ideas without becoming bogged down in small details. As such, they can clearly be strengthened. The strong cover property identified in Sec. 3 guarantees fully optimal Π-complements for all possible views of the main schema $\mathbf{E}$. It is possible to provide a localized strong cover property which only guarantees fully optimal complements for a fixed view $\Pi^{\mathbf{E}}_{\mathbf{W}}$, or for some set of views. The details are a straightforward extension to the results presented, but nevertheless should be elaborated in a future paper. Similarly, an extension of the results of Sec. 4 on quasi-universal Π-complements to equivalences amongst sets of attributes is also a worthwhile topic, although the details of exactly what such an extension would look like remain to be developed.

oarieli@mta.ac.il

Extension of the scope to $\bigvee\Pi$-views: In [15], the $\bigvee\Pi$*-views* of a relational schema, which are defined by sets of projections rather than single projections, are considered as the fundamental sets of views. Since the individual projections which comprise such a view may not join losslessly, this is a genuine extension which often yields smaller complements than the Π-framework. An extension of the results of this paper to the $\bigvee\Pi$-setting would provide useful new results.

General characterization of fully optimal complements: Although this work is an important first step towards the understanding of universality and fully optimal complements, it is clearly limited in comparison to the results on characterization of state invariance and reflection invariance, which hold in very general settings not limited to any particular data model. The next major step for this research program is to seek such a generalized extension, which would proceed in two steps. In the first, the results of this paper would be integrated with the *governing join dependencies* of [15], thus lifting the results from schemata constrained by just FDs to those with more general constraints. In the second step, a more general characterization, independent of any particular data model, would be developed. The exact nature of this characterization remains to be identified.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Atzeni, P., Antonellis, V.D.: Relational Database Theory. Benjamin/Cummings (1993)
3. Bancilhon, F., Spyratos, N.: Independent components of databases. In: Proceedings of the Seventh International Conference on Very Large Data Bases, pp. 398–408 (1981)
4. Bancilhon, F., Spyratos, N.: Update semantics of relational views. ACM Trans. Database Systems 6, 557–575 (1981)
5. Bernstein, P.A.: Synthesizing third normal form relations from functional dependencies. ACM Trans. Database Syst. 1(4), 277–298 (1976)
6. Biskup, J., Dayal, U., Bernstein, P.A.: Synthesizing independent database schemas. In: Proceedings of the ACM-SIGMOD 1979 Conference on Management of Data, pp. 143–151 (1979)
7. Biskup, J., Demetrovics, J., Libkin, L., Muchnik, I.B.: On relational database schemes having unique minimal key. Elektronische Informationsverarbeitung und Kybernetik 27(4), 217–225 (1991)
8. Bogart, K.P.: Introductory Combinatorics. Pitman (1983)
9. Bohannon, A., Pierce, B.C., Vaughan, J.A.: Relational lenses: a language for updatable views. In: Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Chicago, Illinois, USA, June 26-28, pp. 338–347 (2006)
10. Davey, B.A., Priestly, H.A.: Introduction to Lattices and Order, 2nd edn. Cambridge University Press (2002)
11. Foster, J.N., Greenwald, M.B., Moore, J.T., Pierce, B.C., Schmitt, A.: Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. ACM Trans. Programming Languages and Systems 29(3) (2007)

oarieli@mta.ac.il

12. Hegner, S.J.: Foundations of Canonical Update Support for Closed Database Views. In: Kanellakis, P.C., Abiteboul, S. (eds.) ICDT 1990. LNCS, vol. 470, pp. 422–436. Springer, Heidelberg (1990)
13. Hegner, S.J.: An order-based theory of updates for closed database views. Ann. Math. Art. Intell. 40, 63–125 (2004)
14. Hegner, S.J.: The complexity of embedded axiomatization for a class of closed database views. Ann. Math. Art. Intell. 46, 38–97 (2006)
15. Hegner, S.J.: Characterization of Optimal Complements of Database Views Defined by Projection. In: Schewe, K.D., Thalheim, B. (eds.) 4th International Workshop, SDKB 2010, Bordeaux, France. Revised Selected Papers. LNCS, vol. 6384, pp. 73–95. Springer, Heidelberg (2011)
16. Hull, R.: Finitely specifiable implicational dependency families. J. Assoc. Comp. Mach. 31(2), 210–226 (1984)
17. Jacobs, B.E., Aronson, A.R., Klug, A.C.: On interpretations of relational languages and solutions to the implied constraint problem. ACM Trans. Database Systems 7(2), 291–315 (1982)
18. Lechtenbörger, J.: The impact of the constant complement approach towards view updating. In: Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, San Diego, California, June 09-11, pp. 49–55 (2003)
19. Lechtenbörger, J., Vossen, G.: On the computation of relational view complements. ACM Trans. Database Systems 28, 175–208 (2003)
20. Maier, D.: Minimum covers in the relational database model. J. ACM 27(4), 664–674 (1980)
21. Maier, D.: The Theory of Relational Databases. Computer Science Press (1983)
22. Mann, H.B., Ryser, H.J.: Systems of distinct representatives. American Math. Monthly 60(6), 297–401 (1953)
23. Marcus, M.: A Survey of Finite Mathematics. Dover (1993)
24. Paredaens, J., De Bra, P., Gyssens, M., Van Gucht, D.: The Structure of the Relational Database Model. Springer, Heidelberg (1989)
25. Rissanen, J.: Independent components of relations. ACM Trans. Database Systems 2(4), 317–325 (1977)

oarieli@mta.ac.il

Encoding Databases Satisfying a Given Set of Dependencies

Gyula O.H. Katona[1,⋆] and Krisztián Tichler[2,⋆⋆]

[1] Rényi Institute, Budapest, Hungary
ohkatona@renyi.hu
[2] Eötvös University, Faculty of Informatics, Budapest, Hungary
ktichler@inf.elte.hu

Abstract. Consider a relation schema with a set of dependency constraints. A fundamental question is what is the minimum space where the possible instances of the schema can be "stored". We study the following model. Encode the instances by giving a function which maps the set of possible instances into the set of words of a given length over the binary alphabet in a decodable way. The problem is to find the minimum length needed. This minimum is called the information content of the database.

We investigate several cases where the set of dependency constraints consist of relatively simple sets of functional or multivalued dependencies.

We also consider the following natural extension. Is it possible to encode the instances such a way that small changes in the instance cause a small change in the code.

Keywords: relational database, coding, functional dependency, multivalued dependency.

1 Introduction

Let (R, Σ) be a dependency schema in the relational database model ([1]) where R is a relational schema with a single relation and Σ is a set of dependencies on the set of attributes Ω of size $|\Omega| = n$. Suppose that all the domains of the attributes are finite. Then the number of possible tuples is also finite. Hence the number of possible instances I is finite, too. A fundamental question is "what is the minimum space where a database can be stored?". Some of the possible applications are efficient and error-tolerant data transmission or achiving.

Let us repeat the problem for readers not so familiar with the notations above. A database is a table (matrix) of n columns. A row or a record contains the data of one object or person, where the ith element of the row is the ith attribute,

⋆ The work of the first author was supported by the Hungarian National Foundation for Scientific Research grant numbers T037846.

⋆⋆ Research projects presented in this article are supported by the European Union and co-financed by the European Social Fund (grant agreement no. TÁMOP 4.2.1./B-09/1/KMR-2010-0003).

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 203–223, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

the ith type of data of the object. The number of possible values in one place in a row is finite. There are some constraints, rules, connections among the values in a row, given by Σ. There are only finitely many possible rows satisfying these constraints.

An actual situation in the database is a collection of these possible tuples. This is called an instance I of the relation. We need to store the instances in such a way that different instances have different "stored forms". On the other hand the "stored forms" should be relatively small. Our goal is to give a model of this situation.

The following model is suggested. Encode the instances by 0,1 sequences of length ℓ, that is, give a function $c : \mathcal{I}(R, \Sigma) \to \{0,1\}^\ell$ which maps the set of possible instances $\mathcal{I}(R, \Sigma)$ into $\{0,1\}^\ell$. Of course the map should be decodable, c should give different sequences for different instances. $\ell = |c|$ is called the code length. The problem is to find the minimum of ℓ. This minimum can be called the information content of the database schema: $\mathrm{Inf}(R, \Sigma)$. Of course, this is nothing else but the log of the total number of of possible instances, $\lceil \log_2 |\mathcal{I}(R, \Sigma)| \rceil$.

Although the definition is simple and natural, there are difficulties in its implementation. In most cases it is impossible to give an exact number for the total number of instances. We will show a simple-looking example of a trivial multivalued dependency when it is not easy to determine even the asymptotical number of instances.

Our other toy-example is when there is only one minimal key in the dependency schema. In that case we were able to give an exact formula for the number of possible instances using elementary steps. Of course there is a code with length $\mathrm{Inf}(R, \Sigma)$. But this code is useful only when it can be obtained by a simple algorithm and can be similarly decoded, that is the instance can be obtained from the code by another easy algorithm. We do not know if this can be done in our case of only-one-key. However we can show a very natural code which is only slightly longer than $\mathrm{Inf}(R, \Sigma)$.

The next problem arises when the instance is subject to an elementary modification. There is a very natural requirement on these codes. If two instances are similar then their codes should also be similar. More precisely we should write "close" in the previous sentence rather than "similar". If this condition is not satisfied it might happen that making a little change in the database (instance) the changes in the encoded version are big, we have to work too much to get the changes.

Consider some elementary changes in I, like deleting or adding a row, replacing one entry in one of the rows. We would like to have a small change in the code of an instance if it is a subject of one of such elementary changes. The changes in the codes are measured by the Hamming distance that is the number of different digits.

We will show that if this requirement takes place in a fairly strict manner then the code is much longer.

Let us introduce some basic notations that are used in the paper. For an n-tuple $t = (t_{A_1}, \ldots, t_{A_n})$ and $X \subseteq \Omega = \{A_1, \ldots, A_n\}$ let $\pi_X(t)$ denote the

oarieli@mta.ac.il

$|X|$-tuple u that has the property $u_A = t_A$ $(A \in X)$. Sometimes $\pi_X(t)$ is also called an *X-tuple*. If I is an instance let $\pi_X(I) = \{\pi_X(t) \mid t \in I\}$.

For integers r, s $[r]$ denotes the set $\{1, \ldots, r\}$ and $[r, s]$ denotes the set $\{r, \ldots, s\}$.

The paper is organized as follows. In section 2 we consider the case where only one key is given in the schema. The concepts of 2-distance-preserving and strongly 2-distance-preserving codes are introduced. We give lower bounds on the size of these codes in section 4. In section 3 the case of joins is analyzed and an other simple but much different set of multivalued dependencies is considered. Investigations lead to a problem on random bipartite graphs. A partial solution is given in section 5. In section 6 we mention some related works and finish the study with several open problems in section 7.

2 Only One Minimal Key

Let the number of attributes of R be $|\Omega| = n = a + b$ where a and b are positive integers. Suppose that all domains are $\{0, 1\}$. Let the attributes be ordered and suppose that the set $K \subseteq \Omega, |K| = a$ is a key. It can be supposed without loss of generality that K is the set of the first a attributes. Let this dependency schema be denoted by $(R, \{K \to \Omega\})$.

If t is an n-tuple in an instance I satisfying $K \to \Omega$ then $\pi_K(t)$ uniquely determines $\pi_{\Omega - K}(t)$. For an instance $I \in \mathcal{I}(R, \{K \to \Omega\})$ and $u \in \pi_K(I)$ let $f(I, u)$ denote the function describing this dependency, that is $f(I, \pi_K(t)) = \pi_{\Omega - K}(t)$ for an n-tuple t. Of course this function depends on I.

Proposition 1

$$\mathcal{I}(R, \{K \to \Omega\}) = \left(2^b + 1\right)^{2^a}. \tag{1}$$

Proof. The number of possible K-tuples (first part of the n-tuples) is 2^a. Let us denote the ith possible a-tuple by $\hat{i}$ $(0 \leqslant i < 2^a)$. For any given $\hat{i} = \pi_K(t)$ there are 2^b choices for $f(I, \hat{i})$, that is each first part has 2^b possible "continuations" in the last b attributes. If $s = |\pi_K(I)|$ then the total number of possible choices for $f(I, \hat{i})$ for all tuples in the instance is $\left(2^b\right)^s$. There are 2^a possible values of $\pi_K(t)$ therefore one can choose s pieces of K-tuples $\pi_K(t)$ in $\binom{2^a}{s}$ many ways, so the number of instances of size s is

$$\binom{2^a}{s} \left(2^b\right)^s, \tag{2}$$

and the total number of choices is

$$\sum_{s=0}^{2^a} \binom{2^a}{s} \left(2^b\right)^s = \left(2^b + 1\right)^{2^a}. \tag{3}$$

□

Therefore $\mathrm{Inf}(R, \{K \to \Omega\}) = 2^a \log(2^b + 1)$ what is slightly more than $b2^a$. Of course there is always a code c with the length $\ell = \lceil \log(2^b + 1)2^a \rceil$, since we can

oarieli@mta.ac.il

list all the possible instances and the code of the jth instance can be the binary form of j. However nothing ensures that this code is algorithmic and easy to decode.

There is a nice way to encode the instances in the following way. The code $c(I)$ will be determined in the form $c_0(I)c_1(I)\dots c_{2^a}(I)$ where $c_0(I)$ is a 0,1 sequence of length 2^a: its ith digit is 1 iff $\widehat{i} \in \pi_K(I)$ and

$$c_{i+1}(I) = \begin{cases} f(I,\widehat{i}) & \text{if } \widehat{i} \in \pi_K(I) \\ (0,0,\dots,0)(\text{ of length } b) & \text{if } \widehat{i} \notin \pi_K(I) \end{cases} \tag{4}$$

for $0 \leqslant i < 2^a$. The length of this code is $(1+b)2^a$, that is only a little more than $\mathrm{Inf}(R)$. It is easy to see that this code is uniquely decodable, since $c_0(I)$ determines which first parts are in the instance, the second parts are all determined by the function $f(I,\widehat{i})$.

Let us show by a small example why we have to define the second row of (4) to be 0. Choose $a = 2, b = 1$ and consider the instance

$$I = \begin{matrix} 0\,0\,1 \\ 0\,1\,1 \end{matrix}$$

Here $c_0(I)$ is clearly $(1,1,0,0)$ since the first two of the possible four first parts ((0,0), (0,1), (1,0), (1,1)) occur in the instance. $f(I,\widehat{0}) = 1, f(I,\widehat{1}) = 1$ are also well-defined. But $f(I,\widehat{2})$ and $f(I,\widehat{3})$ are not defined by the instance, so all of the following sequences could be codes of the present I: $(1,1,0,0,1,1,0,0)$, $(1,1,0,0,1,1,0,1)$, $(1,1,0,0,1,1,1,0)$, $(1,1,0,0,1,1,1,1)$. To avoid this ambiguity we chose the first one.

We have actually proved the following statement.

Proposition 2. *There is a code of length $\ell = (1+b)2^a$, defined by an easy algorithm that can be easily decoded.*

Deleting the parts in the second line of (4) we obtain a variable length code what is shorter in average, but it is inconvenient otherwise. Therefore define $c^*(I)$ skipping the 0 sequences in the second row of (4).

Proposition 3. *The average length of the code $c^*(I)$ is*

$$\left(1 + b\frac{2^b}{2^b+1}\right)2^a. \tag{5}$$

Proof. The length of $c_0^*(I) = c_0(I)$ does not depend on I, it is 2^a, we can consider the average length of the rest of the code. If $s = |\pi_K(I)|$ then the length of the second part of the code is bs. By (2) the sum of the lengths is

$$\sum_{s=0}^{2^a}\binom{2^a}{s}\left(2^b\right)^s bs = 2^a 2^b b\sum_{s=1}^{2^a}\binom{2^a-1}{s-1}\left(2^b\right)^{s-1} = 2^a 2^b b(2^b+1)^{2^a-1}. \tag{6}$$

oarieli@mta.ac.il

We have to divide this by the number of instances (codewords) what was determined in Proposition 1. The ratio is really

$$2^a b \frac{2^b}{2^b+1}. \tag{7}$$

□

Let us now investigate the condition that small changes cause a small change in the code. Of course it is hopeless to achieve this in the case of codes of variable length. Therefore we return to the codes of fixed length.

3 elementary changes of I will be considered.

(i) Delete one tuple.
(ii) Flip one digit in the first part, $\pi_K(t)$.
(iii) Flip one digit in the second part, $\pi_{\Omega-K}(t)$.

Let us see what the changes are in our code described in Proposition 2. In case (i), if the ith tuple is deleted then $f(I,\widehat{i})$ becomes a sequence of zeros. The number of changes is $b+1$ since a 1 is also replaced by a 0 in c_0. (ii) This change causes 2 changes in c_0 and $2b$ in the rest, altogether $2(b+1)$. Finally, in case (iii) there is only one change in $f(I,\widehat{i})$ if the original change was in the ith tuple.

But can we expect anything better? In case of (ii) we have to give a lot of new information, because the unchanged second part becomes the continuation of another, new first part, that is unrelated to the present one. So, only the changes (i) and (iii) might induce a small change in the code.

(There is an embarrassing question here. Deleting one tuple from the instance seems to be easy, we do not add too much new information. But adding a new tuple needs that. So, one feels that the first change should induce a small change in the code, while the second one does much more. This really happens in the case of the code of variable length given in Proposition 3. But in the case of codes of fixed length the situation is symmetric, the Hamming distance has no direction. This is why we do not treat "adding a new tuple" as an elementary change.)

Can the Hamming distance of the codes of instances obtained by both elementary changes (i) and (iii) be equal to one? Let

$$I_1 = \begin{array}{c|c} 0\,0\ldots0\,0 & 0\,0\ldots0\,0 \\ 0\,0\ldots0\,1 & 0\,0\ldots0\,0 \end{array},$$

$$I_2 = \begin{array}{c|c} 0\,0\ldots0\,0 & 0\,0\ldots0\,0 \\ 0\,0\ldots0\,1 & 0\,0\ldots0\,1 \end{array}$$

and

$$I_3 = 0\,0\ldots0\,0 \,|\, 0\,0\ldots0\,0\,.$$

Deleting the second tuple from I_1 or from I_2, the instance I_3 is obtained. Hence $d(c(I_1),c(I_3)) = 1, d(c(I_2),c(I_3)) = 1$ should hold. On the other hand, I_2 can be obtained from I_1 by flipping one digit in the second part, therefore

oarieli@mta.ac.il

$d(c(I_1), c(I_2)) = 1$. This is a contradiction, since there are no three 0,1 sequences with pairwise Hamming distance 1.

The conclusion is that we can only suppose that the Hamming distance is at most 2 in the cases (i) and (iii). We say that a code $c(I)$ is *2-distance-preserving* if the following two conditions are satisfied.

(iv) If I_2 is obtained from I_1 by deleting one tuple then $d(c(I_1), c(I_2)) \leqslant 2$.
(v) If I_2 is obtained from I_1 by flipping one digit in $\pi_{\Omega-K}(t)$ (the second part) then $d(c(I_1), c(I_2)) \leqslant 2$.

It is easy to construct a code satisfying this condition. Define the following $2^a \times (1+2^b)$ matrix where the rows represent the first part of I. The 0th column contains a 1 in the ith row if $\widehat{i} \in \pi_K(I)$, otherwise it is 0. The ith row contains only 0s if $\widehat{i} \notin \pi_K(I)$. Otherwise the jth entry ($1 \leqslant j \leqslant 2^b$) in the ith row is 1 iff $f(I, \widehat{i})$ is the jth element of the set of all b-tuples, $\{0,1\}^b$ in a given order. Arranging the entries of this matrix in any way a 2-distance-preserving code is obtained. Its length is $2^a(1+2^b)$ what is much larger than $\mathrm{Inf}(R)$ if b is large. A lot is lost as a tradeoff for having the distance preserving property. Call this code the *trivial code* and denote it by c_{tr}. So we have

$$|c_{\mathrm{tr}}| = 2^a(1+2^b). \tag{8}$$

The following bound is exponential in b, unlike the construction of Proposition 2.

Theorem 1. *If c is a 2-distance-preserving code of $(R, \{K \to \Omega\})$ then* $|c| \geqslant \sqrt{2} \cdot 2^{a/2} \cdot 2^{b/2} - 1$.

Observe that c_{tr} has an additional property what cannot be naturally supposed. Namely, changing the second part of any tuple in an instance, the code changes at at most two places:

(vi) If I_2 is obtained from I_1 by changing $\pi_{\Omega-K}(t)$ (the second part) in any arbitrary way then $d(c(I_1), c(I_2)) \leqslant 2$.

Codes satisfying properties (iv) and (vi) are called *strongly 2-distance-preserving*. The following theorem gives an improved bound on the length of such codes. This bound has the right order of magnitude in b.

Theorem 2. *Let $b \geqslant 2$. If c is a strongly 2-distance-preserving code of $(R, \{K \to \Omega\})$ then* $|c| \geqslant 2^b$.

3 A Simple Multivalued Dependency

In this section we consider the information content of multivalued dependency schemas. First, let us remind the reader of the definition of a multivalued dependency (mvd).

oarieli@mta.ac.il

Let R be a relation schema with attribute set Ω and $A, B \subseteq \Omega$. An instance $I = I(R)$ satisfies the *multivalued dependency* $A \twoheadrightarrow B$ if for any tuples $u, v \in I$ and $\pi_A(u) = \pi_A(v)$ implies that there exists a tuple $w \in I$ such that $\pi_A(w) = \pi_A(u) = \pi_A(v)$, $\pi_B(w) = \pi_B(u)$,$\pi_{\Omega-A-B}(w) = \pi_{\Omega-A-B}(v)$ holds.

Note, that due to symmetry reasons, there is also a $w' \in I$, such that $\pi_A(w') = \pi_A(u) = \pi_A(v)$, $\pi_B(w') = \pi_B(v)$,$\pi_{\Omega-A-B}(w') = \pi_{\Omega-A-B}(u)$ holds. Let Σ be a set of mvd's. So, according to the previous more general definition, we can say that an instance $I = I(R)$ satisfies the *multivalued dependency schema (mvd schema)* (R, Σ) if I satisfies all mvd's of Σ. We denote the set of such I's by $\mathcal{I}(R, \Sigma)$.

Let us introduce the following notations for a family of sets $\mathcal{F} = \{X_i \mid 1 \leqslant i \leqslant r\}$. Let

$$\mathcal{M}_{\mathcal{F}} = \left\{ \varnothing \neq X \subseteq \Omega \mid X = \bigcap_{i \in H} X_i \text{ for some } H \subseteq [r] \right\} \tag{9}$$

and

$$a(\mathcal{F}) = \{H \in \mathcal{M}_{\mathcal{F}} \mid \forall H' \subset H : H' \notin \mathcal{M}_{\mathcal{F}}\}. \tag{10}$$

First, let us consider a simple case when $\Sigma = \{\varnothing \twoheadrightarrow X_i \mid X_i \subseteq \Omega, 1 \leqslant i \leqslant r\}$. If $\varnothing \twoheadrightarrow X_i$ holds, then $\varnothing \twoheadrightarrow \Omega - X_i$ holds, too.

Example 1. Let $\Omega = \{\text{Professor, Course, Book}\}$ and consider the following instance I:

Professor	Course	Book
Taylor	Ocean Studies	Corals
Smith	Ocean Studies	Corals
Taylor	Ocean Studies	Whales
Smith	Ocean Studies	Whales
Taylor	Mammals	Whales
Smith	Mammals	Whales
Taylor	Mammals	Monkeys
Smith	Mammals	Monkeys

We have $\varnothing \twoheadrightarrow \{\text{Course}, \text{Book}\}$, since if we have two tuples of I and we exchange the Course-Book pair, the new tuples are in I as well. This can be also viewed as we changed the professor, so $\varnothing \twoheadrightarrow \{\text{Professor}\}$ holds, too. On the other hand $\varnothing \twoheadrightarrow \{\text{Course}\}$ does not hold. Consider the tuples (Taylor, Ocean Studies, Corals) and (Smith, Mammals, Whales), but Prof. Taylor does not teach Mammals from the books on Corals. Note, that $\{\text{Course}\} \twoheadrightarrow \{\text{Book}\}$ holds as well. □

Let $\mathcal{F}_\Sigma = \{X_i \mid 1 \leqslant i \leqslant r\} \cup \{\Omega - X_i \mid 1 \leqslant i \leqslant r\}$. Then clearly $a(\mathcal{F}_\Sigma) = \{S_1, \ldots, S_m\}$ is a partiton of Ω, moreover each $F \in \mathcal{F}$ is the union of some elements of $a(\mathcal{F}_\Sigma)$. Let us introduce the notation $\mathcal{D}_Y = \times_{j \in Y} D_j$ for any $Y \subseteq \Omega$.

oarieli@mta.ac.il

Theorem 3. *Let $D_1, \ldots, D_{|\Omega|}$ be the domains of the attributes of R and $\Sigma = \{\emptyset \twoheadrightarrow X_i \mid X_i \subseteq \Omega, 1 \leqslant i \leqslant r\}$. Then*

$$\mathcal{I}(R, \Sigma) = \{\emptyset\} \cup \{T_1 \times \cdots \times T_m \mid \emptyset \neq T_i \subseteq \mathcal{D}_{S_i}, 1 \leqslant i \leqslant m\}. \tag{11}$$

Proof. Consider a nonempty instance of the type of the right hand side of (11) and an mvd $\emptyset \twoheadrightarrow X_i \in \Sigma$. Since $X_i = \bigcup\{S_j \mid j \in H\}$ and $\Omega - X_i = \bigcup\{S_j \mid j \in [m] - H\}$ holds for some $H \subseteq [m]$ then, according to the definition of direct product, for any tuples u_1 and u_2 of I there exist two tuples v_1 and v_2 such that $\pi_{X_i}(v_1) = \pi_{X_i}(u_1)$, $\pi_{\Omega - X_i}(v_1) = \pi_{\Omega - X_i}(u_2)$, $\pi_{X_i}(v_2) = \pi_{X_i}(u_2)$ and $\pi_{\Omega - X_i}(v_2) = \pi_{\Omega - X_i}(u_1)$ holds, so I satisfies the dependency $\emptyset \twoheadrightarrow X_i$. This holds for each i, so $I \in \mathcal{I}(R, \Sigma)$.

On the other hand consider an instance $I \in \mathcal{I}(R, \Sigma)$ and two tuples u and v. We claim that there is a tuple w in I having the property

$$\pi_{S_1}(w) = \pi_{S_1}(v) \text{ and } \pi_{\Omega - S_1}(w) = \pi_{\Omega - S_1}(u). \tag{12}$$

Since $S_1 \in a(\mathcal{F}_\Sigma)$ we know, that $S_1 = \bigcap \mathcal{F}'$ holds for some $\mathcal{F}' \subseteq \mathcal{F}_\Sigma$. Let $G_i = \bigcap\{F_j \mid 1 \leqslant j \leqslant i\}$ $(0 \leqslant i \leqslant |\mathcal{F}'|)$. There exists a tuple $w_i \in I$ with

$$\pi_{G_i}(w_i) = \pi_{G_i}(v) \text{ and } \pi_{\Omega - G_i}(w_i) = \pi_{\Omega - G_i}(u). \tag{13}$$

In fact, $w_0 = u \in I$ holds and suppose that $w_i \in I$ exists with property (13) for some $0 \leqslant i < |\mathcal{F}'|$. Let us define w_{i+1} by

$$\pi_{F_{i+1}}(w_{i+1}) = \pi_{F_{i+1}}(w_i) \text{ and } \pi_{\Omega - F_{i+1}}(w_{i+1}) = \pi_{\Omega - F_{i+1}}(u). \tag{14}$$

Since $G_{i+1} = G_i \cap F_{i+1}$ w_{i+1} satisfies property (13) (for $i+1$). By induction on i we have $w_i \in I$ and we know that $u \in I$ and $\emptyset \twoheadrightarrow F_{i+1}$ holds for I. Therefore we have $w_{i+1} \in I$ by (14) and the definition of mvd.

Setting w to be $w_{|\mathcal{F}'|}$ proves the claim since w has property (12).

Analogously, for each $1 \leqslant i \leqslant m$ there exist a tuple $w_{\{S_i\}} \in I$ having the property

$$\pi_{S_i}(w_{\{S_i\}}) = \pi_{S_i}(v) \text{ and } \pi_{\Omega - S_i}(w_{\{S_i\}}) = \pi_{\Omega - S_i}(u). \tag{15}$$

We prove by induction on $|\mathcal{H}|$, that there exists a tuple $w_{\mathcal{H}} \in I$ for any $\mathcal{H} \subseteq a(\mathcal{F}_\Sigma)$ satisfying

$$\pi_{\bigcup \mathcal{H}}(w_{\mathcal{H}}) = \pi_{\bigcup \mathcal{H}}(v) \text{ and } \pi_{\Omega - \bigcup \mathcal{H}}(w_{\mathcal{H}}) = \pi_{\Omega - \bigcup \mathcal{H}}(u). \tag{16}$$

For $|\mathcal{H}| \leqslant 1$ the statement holds, so let $|\mathcal{H}| \geqslant 2$. Suppose, that it is true for any $\mathcal{H}' \subseteq a(\mathcal{F}_\Sigma)$ such that $|\mathcal{H}'| < |\mathcal{H}|$. There exists an $F \in \mathcal{F}_\Sigma$ satisfying $F \cap \bigcup \mathcal{H} \neq \emptyset$ and $F \cap \bigcup \mathcal{H} \neq \bigcup \mathcal{H}$, otherwise $a(\mathcal{F}_\Sigma)$ contains $\bigcup \mathcal{H}$ or a superset of it, but this is not possible for $|\mathcal{H}| \geqslant 2$. $F \cap \bigcup \mathcal{H}$ and $\bigcup \mathcal{H} - F$ are disjoint unions of some S_i's, fewer than $|\mathcal{H}|$. By the induction hypothesis there exists $w_{F \cap \bigcup \mathcal{H}} \in I$ and $w_{\bigcup \mathcal{H} - F} \in I$ satisfying (16) (for $F \cap \bigcup \mathcal{H}$ and $\bigcup \mathcal{H} - F$ respectively). But then since $\emptyset \twoheadrightarrow F$ holds for I, $w_{\bigcup \mathcal{H}} \in I$ satisfying (16) exists.

So any $I \in \mathcal{I}(R, \Sigma)$ is of the type of the right hand side of (11). □

oarieli@mta.ac.il

Corollary 1

$$|\mathcal{I}(R,\Sigma)| = 1 + \prod_{S_i \in a(\mathcal{F}_\Sigma)} \left(2^{\prod_{j\in S_i}|D_j|} - 1\right) \tag{17}$$

□

We suggest the following coding of the instances. The code consists of $m(= |a(\mathcal{F}_\Sigma)|)$ blocks. The ith block has length $\prod_{j\in S_i}|D_j|$, each bit corresponds to an element of $\times_{j\in S_i} D_j$. Note, that only those codewords that have all 0's in a particular block are not used, except for the all 0 codeword.

By corollary 1 the length of this code is the best possible if

$$2\left(1 + \prod_{S_i \in a(\mathcal{F}_\Sigma)} \left(2^{\prod_{j\in S_i}|D_j|} - 1\right)\right) > \prod_{S_i \in a(\mathcal{F}_\Sigma)} \left(2^{\prod_{j\in S_i}|D_j|}\right) \tag{18}$$

holds. Note that there are cases in which (18) does not hold, typically in the case where the cardinality of the domains are small.

Coding and decoding is easily computable.

There is a natural partial ordering on $\mathcal{I}(R,\Sigma)$, namely let $T_1 \times \ldots \times T_m \preccurlyeq T'_1 \times \cdots \times T'_m$ if $T_i \subseteq T'_i$ holds for all $1 \leqslant i \leqslant m$ ($\emptyset$ is smaller than anything).

Let us consider various updates on instance I. Note, that the modified instance I' should be in $\mathcal{I}(R,\Sigma)$ as well. So restoring the dependency schema maybe neccessary. First, we define quite natural ways to restore the schema, then we investigate the changes in the codeword of I. (Of course there is always a natural way to restore the schema: undo. But we exclude this way.)

Tuple deletion: To restore the dependencies after deleting u let the resulting instance be such an instance I', that satisfiy $I' \preccurlyeq I$, that does not contain u and that is maximal in the partial ordering with this property. This can be done by modifying a single bit in the codeword, unless I consists of a single tuple.

Tuple insertion: To restore the dependencies after inserting u let the resulting instance be such an instance I', that satisfiy $I \preccurlyeq I'$, that does contain u and that is minimal in the partial ordering with this property. This can be done by modifying as many bits in the codeword as "new" $\pi_{T_i}(u)$'s are among the corresponding projections of u.

Entry modification: To restore the dependencies after a modification in the attribute $j(\in S_i)$ of u to u' (i.e., $\pi_i(u) \neq \pi_i(u') \Leftrightarrow i = j$ $(1 \leqslant i \leqslant |\Omega|)$) we have 2 cases. Either $\pi_{S_i}(u') \in T_i$ (that is the modified tuple is the same as one the other tuples of I) or not (the modifed tuple was not a tuple of I yet). In the first case, let I' be such an instance, that satisfies $I' \preccurlyeq I$, that does not contain u and that is maximal in the partial ordering with this property. For the the second case we use I'' determined by the first case. Let I' be such an instance that satisfiy $I'' \preccurlyeq I'$, that does contain u' and that is minimal in the partial ordering with this property. In the first case this corresponds to a modification of a single bit in the ith block (see tuple deletion), in the second case 2 bits are modified in the same block.

This natural code is quite good in the sense that small changes in the instance result in small changes in the code.

oarieli@mta.ac.il

What can we say if the left hand sides of the mvd's are not the empty sets, but say X (the same for all mvd's)? Since $X \twoheadrightarrow Y$ implies $X \twoheadrightarrow Y - X$ and vica versa we can consider $\Sigma'' = \{X \twoheadrightarrow X_i - X \mid X_i \subseteq \Omega, 1 \leqslant i \leqslant r\}$ instead of $\Sigma' = \{X \twoheadrightarrow X_i \mid X_i \subseteq \Omega, 1 \leqslant i \leqslant r\}$. The instances satisfying Σ'' are

$$\{\varnothing\} \cup \left\{ \bigcup_{w \in \mathcal{H}} \{w\} \times F(w) \,\middle|\, \varnothing \neq \mathcal{H} \subseteq \mathcal{D}_X, F : \mathcal{H} \to \bigtimes_{i=1}^{m} \left(2^{\mathcal{D}_{S_i}} - \{\varnothing\}\right) \right\}. \quad (19)$$

In other words, for a given instance I satisfying (R, Σ'') and an X-tuple $w \in \pi_X(I)$ the continuations that complete w to an n-tuple of I are of the form (11) (for $\Omega - X$ instead of Ω). The proof of (19) is analogous to the proof of Theorem 3.

This similarity motivates to study the case of a simple set of mvd's that looks completely different. Let $\Sigma = \{A \twoheadrightarrow C, B \twoheadrightarrow D\}$, where $B = C - A$ and $A = D - B$. If these 2 conditions hold, Σ is equivalent to the set $\{A \twoheadrightarrow B, B \twoheadrightarrow A\}$ (since $A \twoheadrightarrow C$ implies $A \twoheadrightarrow C - A$ and vica versa) or more simply to the situation where $\Omega = \{1, 2, 3\}$ and $\Sigma = \{\{1\} \twoheadrightarrow \{2\}, \{2\} \twoheadrightarrow \{1\}\}$. Let $n = |D_1|, m = |D_2|, k = |D_3|$.

For any $I \subseteq D_1 \times D_2 \times D_3$ let $G = G_I = (D_1, D_2, E)$ be the bipartite graph, that has $\{x, y\} \in E(G) \Leftrightarrow \exists z \in D_3\ (x, y, z) \in I$. Furthermore for an edge $e = \{x, y\} \in E(G)$ let $S_e = \{z \in D_3 \mid (x, y, z) \in I\}$.

Lemma 1. *Let $\Omega = \{1, 2, 3\}$ and $\Sigma = \{\{1\} \twoheadrightarrow \{2\}, \{2\} \twoheadrightarrow \{1\}\}$. $\varnothing \neq I \in \mathcal{I}(R, \Sigma)$ if and only if for any $e, f \in E(G_I)$, that are in the same connected component of G $S_e = S_f$ holds.*

Proof. Suppose that for any $e, f \in E(G_I)$, that are in the same component of G $S_e = S_f$ holds. Consider two tuples (x, y, z) and (x, y', z'). Since $\{x, y\}$ and $\{x, y'\}$ have a common vertex they are in the same component. So $S_{\{x,y\}} = S_{\{x,y'\}}$, which implies $(x, y, z'), (x, y', z) \in I$, so $\{1\} \twoheadrightarrow \{2\}$ holds. Similarly $\{2\} \twoheadrightarrow \{1\}$ holds as well, so $I \in \mathcal{I}(R, \Sigma)$.

On the other hand if $I \in \mathcal{I}(R, \Sigma)$ and $e, f \in E(G_I)$ are in the same edge-connected component then there exists a path $v_0, e_1, v_1, e_2, \ldots v_{\ell-1}, e_\ell, v_\ell$ in G such that $e_1 = e$ and $e_\ell = f$. We can suppose w.l.o.g., that $v_0 \in D_1$. Let z be an arbitrary element of S_e, i.e., $(v_0, v_1, z) \in I$. $\{v_1, v_2\} \in E(G)$ implies that there exist some $z' \in D_3$, such that $(v_2, v_1, z') \in I$. $\{2\} \twoheadrightarrow \{1\}$ implies $(v_2, v_1, z) \in I$. By a similar arguement $\{1\} \twoheadrightarrow \{2\}$ implies $(v_2, v_3, z) \in I$. By easy induction $(v_{\ell-1}, v_\ell, z) \in I$, (or $(v_\ell, v_{\ell-1}, z) \in I$,) which implies $z \in S_f$. Conversely, $z \in S_f$ implies $z \in S_e$, therefore we have $S_e = S_f$. □

Let $c_{n,m,s}$ $(1 \leqslant s \leqslant \min\{n, m\})$ denote the number of bipartite graphs with partition sizes n, m and exactly s connected components that contain at least one edge (so isolated vertices do not count as a component).

oarieli@mta.ac.il

Corollary 2. *Let* $\Omega = \{1,2,3\}$, $n = |D_1|$, $m = |D_2|$, $k = |D_3|$ *and* $\Sigma = \{\{1\} \twoheadrightarrow \{2\}, \{2\} \twoheadrightarrow \{1\}\}$.

$$|\mathcal{I}(R,\Sigma)| = 1 + \sum_{s=1}^{\min\{n,m\}} c_{n,m,s} \left(2^k - 1\right)^s . \tag{20}$$

□

The following bounds on $\mathrm{Inf}(R,\Sigma)$ follow from Corollary 2.

Corollary 3. *Let* $\Omega = \{1,2,3\}$, $n = |D_1| > 0$, $m = |D_2| > 0$, $k = |D_3| > 0$ *and* $\Sigma = \{\{1\} \twoheadrightarrow \{2\}, \{2\} \twoheadrightarrow \{1\}\}$.

$$nm + k - 1 \leqslant \mathrm{Inf}(R,\Sigma) \leqslant nm + \min\{n,m\}k \tag{21}$$

Proof. We have by (20)

$$2^{nm+k-1} \leqslant 1 + (2^{nm}-1)(2^k-1) = 1 + \left(\sum_{s=1}^{\min\{n,m\}} c_{n,m,s}\right)(2^k-1) =$$

$$1 + \sum_{s=1}^{\min\{n,m\}} c_{n,m,s}(2^k-1) \leqslant |\mathcal{I}(R,\Sigma)| \leqslant 1 + \sum_{s=1}^{\min\{n,m\}} c_{n,m,s}(2^k-1)^{\min\{n,m\}} =$$

$$1 + \left(\sum_{s=1}^{\min\{n,m\}} c_{n,m,s}\right)(2^k-1)^{\min\{n,m\}} = 1 + (2^{nm}-1)(2^k-1)^{\min\{n,m\}} \leqslant$$

$$2^{nm+\min\{n,m\}k}. \quad \square \tag{22}$$

For small s we prove some asymptotic bounds on $c_{n,m,s}$ in section 5. We also discuss a conjecture on $c_{n,m,s}$ for all s in section 7. Note, that if Conjecture 2 is true both the lower and the upper bounds of (21) can be close for various n, m, k. These possibilities are discussed in section 7.

Let us consider again the updates of $I \in \mathcal{I}(R,\Sigma)$ and the poset $\langle \mathcal{I}(R,\Sigma), \subseteq \rangle$. Note, that restoring the mvd schema after insertion of a tuple t may result a big change in I if $\pi_{\{1,2\}}(t)$ is a new edge that connects components $C_1, C_2 \subseteq D_1 \times D_2$. If $S_1, S_2 \in 2^{D_3}$ are the subsets of D_3 belonging to C_1 and C_2 respectively, then the own set of the unified component will be $S_1 \cup S_2 \in 2^{D_3}$. So, maybe there is no positive answer for Problem 3 in section 7.

4 Bounds on the Size of 2-Distance-Preserving Codes

Define the graph $G_1(a,b) = (V, E_1)$ where V is the set of all instances of $(R, \{K \to \Omega\})$. Therefore, by Proposition 1 we have $|V| = (2^b+1)^{2^a}$. Two vertices are joined by an edge in $G_1(a,b)$ if the corresponding instances can be obtained by a change (i) or (iii). B_ℓ is the ℓ-dimensional cube, more precisely it is the graph with vertex set $\{0,1\}^\ell$ where two vertices are joined by an edge if their Hamming distance is one. $B_\ell^{\leqslant 2}$ is a graph with the same vertex set, but two vertices are adjacent if their Hamming distance is 1 or 2.

oarieli@mta.ac.il

Let $H_1 = (U_1, F_1)$ and $H_2 = (U_2, F_2)$ be two graphs. An injective map $m : U_1 \to U_2$ is called an *embedding* if the edges in F_1 are mapped into edges in F_2. In notation: $m : H_1 \hookrightarrow H_2$. The following is true by the definitions,

$$c \text{ is a 2-distance-preserving code of length } \ell \text{ iff } c : G_1(a,b) \hookrightarrow B_\ell^{\leqslant 2}.$$

The smallest ℓ for which an embedding $H = (U, F) \hookrightarrow B_\ell^{\leqslant 2}$ exists is called the $(\leqslant 2)$-*dimension* of H. In notation: $\dim^{\leqslant 2}(H)$.

Lemma 2. *Let $m : H = (U, F) \hookrightarrow B_\ell^{\leqslant 2}$ be an embedding and $u \in U$ an arbitrary vertex. Then there is another embedding $m' : H \hookrightarrow B_\ell^{\leqslant 2}$ which maps u to the all-zero sequence.*

Proof. Add $m(u)$ to all the vectors mod 2, that is $m'(v) = m(v) + m(u)$. This operation does not change the Hamming distance, the modified map is also injective, maps edge to edge and $m'(u) = 2m(u)$ which is zero mod 2. □

Lemma 3. *If H_2 is a subgraph of H_1 then $dim^{\leqslant 2}(H_2) \leqslant dim^{\leqslant 2}(H_1)$.*

Proof. Let $m : H_1 \hookrightarrow B_\ell^{\leqslant 2}$ be an embedding where $\ell = \dim^{\leqslant 2}(H_1)$. Then $m|_{H_2}$ is also an embedding of H_2 into the same $B_\ell^{\leqslant 2}$. So by the definition of $(\leqslant 2)$-dimension $\dim^{\leqslant 2}(H_2) \leqslant \ell = \dim^{\leqslant 2}(H_1)$ holds. □

Let K_r be the complete graph on r vertices.

Lemma 4. *$dim^{\leqslant 2}(K_r) = r - 1$ $(r \neq 4)$, $dim^{\leqslant 2}(K_4) = 2$.*

Proof. First we give an embedding. The image of the map are those sequences of length $r - 1$ having at most one 1. The distance of every pair is $\leqslant 2$. Since $B_2^{\leqslant 2}$ is isomorphic to K_4 the set of all 0,1 sequences of length 2 gives a better construction for $r = 4$.

Conversely suppose that $m : K_r \hookrightarrow B_\ell^{\leqslant 2}$ is an embedding. The trivial inequality $\ell \geqslant \lceil \log r \rceil$ proves the correct lower bound for $r = 1, 2, 3, 4$. Suppose $r > 4$.

By Lemma 2 it can be supposed that one of the vertices, say u is mapped to the all-zero sequence of length ℓ. The maps of all other vertices must contain one or two 1s. Let $n(1)$ and $n(2)$ denote the number of sequences containing one and two 1s, respectively. Several cases will be distinguished.

$n(1) \geqslant 3$. The vertices u_1, u_2, u_3 have a map containing exactly one 1. Suppose that the map of a vertex v contains two 1s. Then the Hamming distance of v and at least one of u_1, u_2, u_3 is $\geqslant 3$ what is a contradiction. Therefore $n(1) = r - 1$, and $\ell \geqslant r - 1$, this case is settled.

$n(1) = 2$. Let $m(u_1)$ and $m(u_2)$ contain one 1. If the map $m(v)$ contains two 1s then the place of the 1 in $m(u_1)$ must be one of the two places of 1s in $m(v)$, otherwise their Hamming distance is 3. The same is true for $m(u_2)$ therefore the two places are uniquely determined in $m(v)$. Since $r > 4$ there must be a map whose 1s are not at these two places, contradicting the conditions. This case is impossible.

oarieli@mta.ac.il

$n(1) = 1$. Suppose that the map of u_1 has only one 1, say in the first place in the sequence. The maps with two 1s must have one of them in the first place, again. Their second 1s must occupy $r-2$ different places. Hence we have $\ell \geqslant r-1$.

$n(1) = 0$. All maps have two 1s. Their places in $m(u_1)$ and $m(u_2)$ cannot be 4 distinct ones: this would imply Hamming distance 4. Without loss of generality one can suppose that $m(u_1)$ has 1s in the first and second places, while $m(u_2)$ in the first and third places. If $m(v)$ has 1 at the second and third places, then we cannot choose a good map for the fourth vertex of K_r. Therefore the map of all vertices has a 1 in the first place. The $r-1$ maps (different from the all-zero) occupy r places. In this case $\ell \geqslant r$ is obtained, better than our need. □

Lemma 5. *If $H = (U, F)$ is a graph with one vertex of maximum degree $|U|-1$ then the smallest integer satisfying*

$$1 + x + \binom{x}{2} \geqslant |U| \tag{23}$$

is a lower bound on $dim^{\leqslant 2}(H)$.

Proof. Let $m : H \hookrightarrow B_\ell^{\leqslant 2}$ be an embedding and let u be the vertex with degree $|U| - 1$. By Lemma 2 one can suppose that $m(u)$ is the all-zero sequence. The maps of all other vertices have one or two 1s in their maps. Of course $|U| = 1 + n(1) + n(2)$ holds. Now $n(1) \leqslant \ell$ and $n(2) \leqslant \binom{\ell}{2}$ imply

$$1 + \ell + \binom{\ell}{2} \geqslant |U|. \tag{24}$$

□

Lemma 6. *$G_1(a, b)$ contains a vertex of degree $2^a 2^b$.*

Proof. Consider the empty instance (no tuple) as a vertex of G_1. All the instances with one tuple are neighbors in G_1 since the deletion of the only tuple leads to the empty instance. The number of such tuples is 2^{a+b}. □

Proof (of Theorem 1). Take the subgraph H of G_1 spanned by the empty instance and its neighbors. The number of vertices of H is $1 + 2^a 2^b$. Applying Lemma 5 for H condition (23) becomes $\frac{x(x+1)}{2} \geqslant 2^a 2^b$. The inequality $\frac{(x+1)^2}{2} \geqslant 2^a 2^b$ has smaller or equal solutions. Hence we have $\dim^{\leqslant 2}(H) \geqslant \sqrt{2} 2^{\frac{a}{2}} 2^{\frac{b}{2}} - 1$. By Lemma 3 this is true for G_1, too. □

Define now the graph $G_2(a, b) = (V, E_1)$ that has the same vertex set as $G_1(a, b)$ that is $\mathcal{I}(R, \{K \to \Omega\})$. Two vertices are joined by an edge in $G_2(a, b)$ if the corresponding instances can be obtained by a change (i) or (vi). Now we want to give a lower bound on $\dim^{\leqslant 2}(G_2)$.

Lemma 7. *$G_2(a, b)$ contains a complete subgraph of size $1 + 2^b$.*

oarieli@mta.ac.il

Proof. Take the empty instance and all the instances having one tuple with a fixed first part (say $\pi_K(t) = (1, 0, \ldots, 0, 0)$) and all possible second parts. The number of these instances is really $1 + 2^b$, and any two of them are joined by an edge in G_2. □

Proof (of Theorem 2). Use Lemma 4 with the complete subgraph K_r obtained in Lemma 7 where $r = 1 + 2^b$. Lemma 4 gives $\dim^{\leqslant 2}(K_r) \geqslant 2^b$. Lemma 3 completes the proof. □

5 Asymptotic Bounds on $c_{n,m,s}$

In this section we focus on how to calculate $c_{n,m,s}$. The question is equivalent to counting the probability $p_{n,m,s}$ of a random bipartite graph with partition sizes n, m having exactly s connected components that contain at least one edge.

$$p_{n,m,s} = \frac{c_{n,m,s}}{2^{nm}}. \tag{25}$$

Instead of considering the model of taking graphs with probability $1/2^{nm}$ we can consider the random graph model $G(n, m, 1/2)$, where each edge of the complete bipartite graph $K_{n,m}$ with partition sizes n and m has probability $1/2$ to be included in a random bipartite graph with partition sizes n and m. This way, each random graph will be equally probable, too. In most of the following calculations, we consider the more general $G(n, m, p)$ model, where each edge of the complete bipartite graph $K_{n,m}$ has probability p to be included, and probability $q = 1 - p$ to be not included in the random graph.

We know, that most of the graphs are connected, and that the probability of being not connected is exponentially small. But in (20) the less probable an event is the bigger is its weight, and the weight is exponential, so we need precise counting.

First of all, we need an estimate with error terms for the probability of a random graph being connected. Let us denote this probability by $p'_{n,m}$. Note, that $p'_{n,m} \neq p_{n,m,1}$, since in the latter case isolated vertices are allowed. E. N. Gilbert ([6]) has determined asymptotically the number of disconnected graphs on n vertices. We adapt the ideas for bipartite graphs, but we need more precise estimates. Throughout the counting we suppose that m, n is large enough if the bounds do not hold for some small values.

For the lower bound on $1 - p'_{n,m}$ we can say, that those graphs that have an isolated vertex are surely disconnected. So let E_i be the event that the ith vertex is isolated. Then by Bonferroni's inequality ([5])

$$1 - p'_{n,m} \geqslant \sum_{i=1}^{n+m} \mathrm{P}(E_i) - \sum_{i<j} \mathrm{P}(E_i E_j) =$$
$$nq^m + mq^n - \binom{n}{2} q^{2m} - \binom{m}{2} q^{2n} - nmq^{m+n-1}. \tag{26}$$

oarieli@mta.ac.il

For the upper bound on $1 - p'_{n,m}$ we use the following recursion:

$$1 - p'_{n,m} = \sum_{(i,j)\in\Gamma} \binom{n}{i}\binom{m-1}{j-1} p'_{i,j} q^{i(m-j)+j(n-i)}, \tag{27}$$

where $\Gamma = [0,n] \times [0,m] - \{(n,m)\} - \{(i,0) \mid i \in [n]\} - \{(0,j) \mid 2 \leqslant j \leqslant m\}$.

Let us separate the main terms of the right hand side of (27),

$$\begin{aligned} p'_{0,1}q^n + p'_{1,1}nq^{n+m-2} + p'_{n,m-1}(m-1)q^n + p'_{n-1,m}nq^m + \\ p'_{n-2,m}\frac{n(n-1)}{2}q^{2m} + p'_{n-1,m-1}n(m-1)q^{n+m-2} + \\ p'_{n,m-2}\frac{(m-1)(m-2)}{2}q^{2n} + \mathrm{err}_{n,m}\,. \end{aligned} \tag{28}$$

So $\mathrm{err}_{n,m}$ is defined by (28) equals to the right hand side of (27). Note, that $p'_{0,1} = 1$ and $p'_{1,1} = p$.

An upper bound on $\mathrm{err}_{n,m}$ is the following:

$$\mathrm{err}_{n,m} \leqslant \sum_{(i,j)\in\Gamma_1} \binom{n}{i}\binom{m-1}{j-1} q^{i(m-j)+j(n-i)}, \tag{29}$$

where $\Gamma_1 = \{(i,j) \in [1,n] \times [1,m] \mid 3 \leqslant i+j \leqslant m+n-3\}$.

For an upper estimate on $\mathrm{err}_{n,m}$ we need a lower bound on the exponent of q. Let

$$\begin{aligned} f_1(x) = (n-2)x + m, \quad f_2(x) = \frac{n}{2}\left(x + \frac{m-n}{2}\right), \\ f_3(x) = \frac{n}{2}\left(-x + m + \frac{m+n}{2}\right), \quad f_4(x) = n(-x+m+n). \end{aligned} \tag{30}$$

We suppose for the rest of the proof, that $n \leqslant m$ holds. The proofs of the following elementary inequalities are left to the reader.

$$i(m-j) + j(n-i) \geqslant \begin{cases} f_1(i+j) & 3 \leqslant i+j \leqslant \frac{m-n}{2},\ i \neq 0 \\ f_2(i+j) & \frac{m-n}{2} \leqslant i+j \leqslant \frac{m+n}{2} \\ f_3(i+j) & \frac{m+n}{2} \leqslant i+j \leqslant m+n-\frac{m-n}{2} \\ f_4(i+j) & m+n-\frac{m-n}{2} \leqslant i+j \leqslant m+n-3 \end{cases}. \tag{31}$$

So an upper bound on $\mathrm{err}_{n,m}$ is the following:

$$\begin{aligned} \sum_{t=1}^{4} \sum_{(i,j)\in\Gamma_1} \binom{n}{i}\binom{m-1}{j-1} q^{f_t(i+j)} = \\ \sum_{t=1}^{4} \left(\sum_{(i,j)\in\Gamma_2} \binom{n}{i}\binom{m-1}{j-1} q^{f_t(i+j)} - \sum_{j=1}^{n} \binom{m-1}{j-1} q^{f_t(j)} \right), \end{aligned} \tag{32}$$

oarieli@mta.ac.il

where $\Gamma_2 = [n] \times [1,m] - \{(1,1)\} - \{(i,j) \mid m-i-j \leqslant 2\}$. So for each $1 \leqslant t \leqslant 4$:

$$\sum_{(i,j)\in\Gamma_2} \binom{n}{i}\binom{m-1}{j-1} q^{f_t(i+j)} = \sum_{(i,j)\in[n]\times[1,m]} \binom{n}{i}\binom{m-1}{j-1} q^{f_t(i+j)} - nq^{f_t(2)} - q^{f_t(n+m)} - (n+m-1)q^{f_t(n+m-1)} - \frac{(n+m-1)(n+m-2)}{2} q^{f_t(n+m-2)}. \tag{33}$$

Let $f_t(x) = a_t(x) + c_t$ $(1 \leqslant t \leqslant 4)$, then

$$\sum_{(i,j)\in[n]\times[1,m]} \binom{n}{i}\binom{m-1}{j-1} q^{f_t(i+j)} - \sum_{j=1}^{m} \binom{m-1}{j-1} q^{f_t(j)} = q^{f_t(1)} \left(\sum_{h=0}^{m+n-1} \binom{m+n-1}{h} q^{a_t h} - \sum_{j'=0}^{m-1} q^{a_t j'} \right). \tag{34}$$

Then $\mathrm{err}_{n,m} \leqslant \mathrm{err}_{n,m,1} + \mathrm{err}_{n,m,2} + \mathrm{err}_{n,m,3} + \mathrm{err}_{n,m,4}$, where

$$\mathrm{err}_{n,m,1} = q^{m+n-2}\left((1+q^{n-2})^{n+m-1} - (1+q^{n-2})^{m-1}\right) - nq^{m+2(n-2)} - q^{(n-2)(m+n-2)+m}\left(\frac{(m+n-1)(m+n-2)}{2} + (m+n-1)q^{n-2} + q^{2(n-2)}\right), \tag{35}$$

$$\mathrm{err}_{n,m,2} = q^{\frac{n}{2}(1+\frac{m-n}{2})}\left((1+q^{\frac{n}{2}})^{n+m-1} - (1+q^{\frac{n}{2}})^{m-1}\right) - nq^{\frac{n}{2}(2+\frac{m-n}{2})} - q^{\frac{n}{2}(m+n-2+\frac{m-n}{2})}\left(\frac{(m+n-1)(m+n-2)}{2} + (m+n-1)q^{\frac{n}{2}} + q^{n}\right), \tag{36}$$

$$\mathrm{err}_{n,m,3} = q^{\frac{n}{2}(m-1+\frac{m+n}{2})}\left((1+q^{-\frac{n}{2}})^{n+m-1} - (1+q^{-\frac{n}{2}})^{m-1}\right) - nq^{\frac{n}{2}(m-2+\frac{m+n}{2})} - q^{\frac{n}{2}(2+\frac{m-n}{2})}\left(\frac{(m+n-1)(m+n-2)}{2} + (m+n-1)q^{-\frac{n}{2}} + q^{-n}\right), \tag{37}$$

$$\mathrm{err}_{n,m,4} = q^{n(m+n-1)}\left((1+q^{-n})^{n+m-1} - (1+q^{-n})^{m-1}\right) - nq^{n(m+n-2)} - 1 - (m+n-1)q^{n} - \frac{(m+n-1)(m+n-2)}{2} q^{2n}. \tag{38}$$

This gives an upper bound of

$$\mathrm{err}_{n,m} \leqslant (n+m)^4 \left(q^{3n} + q^{\frac{n}{2}(3+\frac{m-n}{2})} + q^{m+2(n-2)}\right) \tag{39}$$

for $q < (1/(m+n))^{2/n}$ (which holds for large m, n). This gives

$$\mathrm{err}_{n,m} \leqslant \left(\frac{n+m}{q}\right)^4 q^{3n} \tag{40}$$

oarieli@mta.ac.il

for $m - n \geqslant 6$. Note, that by more careful, but similar counting one can show, that (40) is valid in the remaining cases, too. We skip this counting, here. Note, that this is a good bound only if we assume $m < 1.999n$ to ensure that a term of q^{n+m} has smaller exponent than q^{3n}. Having too unbalanced partition sizes would complicate counting, since there is an existing situation in this case, when having 3 (or more) isolated vertices in the larger partition is more probable than having a component with a single edge.

So we proved the following lemma.

Lemma 8. *Let $n \leqslant m$. Then the following holds.*

$$nq^m + mq^n - \binom{n}{2}q^{2m} - \binom{m}{2}q^{2n} - nmq^{m+n-1} \leqslant 1 - p'_{n,m} \leqslant nq^m + mq^n + npq^{n+m-2} + \frac{n(n-1)}{2}q^{2m} + n(m-1)q^{n+m-2} + \frac{(m-1)(m-2)}{2}q^{2n} + \left(\frac{n+m}{q}\right)^4 q^{3n}. \quad (41)$$

Now let us turn our attention to $p_{n,m,1}$. We can have an upper bound on $p_{n,m,1}$ similarly to $p'_{n,m}$. A component with a single edge guarantees that we don't have an edge-connected bipartite graph. We use Bonferroni's inequality ([5]) again.

$$p_{n,m,1} \leqslant 1 - nmpq^{m+n-2} + n(n-1)m(m-1)p^2q^{2n+2m-6} \quad (42)$$

For the upper bound observe, that an edge-connected bipartite graph is such a graph that have i isolated vertices in partition D_1 and j isolated vertices in partition D_2 and the rest of the graph is connected. So

$$p_{n,m,1} = \sum_{i,j} \binom{n}{i}\binom{m}{j} q^{im+jn-ij} p'_{n-i,m-j}. \quad (43)$$

A lower bound is just a few terms of the left hand side of (43), so we have by (28) and (40)

$$p_{n,m,1} \geqslant 1 - (1 + (m-1)p'_{n,m-1})q^n - npq^{n+m-2} - p'_{m,n-1}nq^m - p'_{n-2,m}\frac{n(n-1)}{2}q^{2m} - p'_{n-1,m-1}n(m-1)q^{n+m-2} - p'_{n,m-2}\frac{(m-1)(m-2)}{2}q^{2n} - \mathrm{err}_{n,m} + mq^n p'_{n,m-1} + nq^m p'_{n-1,m} + \binom{m}{2}q^{2n}p'_{n,m-2} + nmq^{m+n-1}p'_{n-1,m-1} + \binom{n}{2}q^{2m}p_{n-2,m} \geqslant 1 - nmpq^{n+m-2} + q^n(p'_{n,m-1} - 1 + (m-1)q^n p'_{n,m-2} + np'_{n-1,m-1}q^{m-2}) \geqslant 1 - nmpq^{n+m-2} - 2\left(\frac{n+m}{q}\right)^4 q^{3n}. \quad (44)$$

We obtain the following by (42) and (44).

oarieli@mta.ac.il

Lemma 9. *Let $n \leqslant m$ Then*

$$nmpq^{m+n-2} - n(n-1)m(m-1)p^2q^{2n+2m-6} \leqslant 1 - p_{n,m,1} \leqslant nmpq^{n+m-2} + 2\left(\frac{n+m}{q}\right)^4 q^{3n}. \quad (45)$$

So we have

$$p_{n,m,2} \leqslant nmpq^{n+m-2} + 2\left(\frac{n+m}{q}\right)^4 q^{3n}. \quad (46)$$

For larger s the situation becomes more complicated. We can have the following lower bound for $p_{n,m,\geqslant s+1}$ by Bonferroni's inequality ([5])

$$p_{n,m,\geqslant s+1} \geqslant \beta_{n,m,s} - \sum_{i=s+1}^{\min\{2s,n,m\}} \beta_{n,m,i}, \quad (47)$$

A lower bound on $p_{n,m,s}$ can be obtained by the equality $p_{n,m,s} = p_{n,m,\geqslant s} - p_{n,m,\geqslant s+1}$.

All bipartite graphs with exactly/at least s component of size at least 2 have a cut to an edge-connected graph with at least an edge and a bipartite graph with exactly/at least $s-1$ component of size at least 2, so we have

$$p_{n,m,s} \leqslant \sum_{(i,j)\in[1,n-s+1]\times[1,m-s+1]} \binom{n}{i}\binom{m}{j} q^{i(m-j)+j(n-i)} p_{i,j,1} p_{n-i,m-j,s-1}. \quad (48)$$

We use induction to upper estimate $p_{n,m,s}$. For $s \geqslant 3$ the main term is when $i = j = 1$ ($p_{1,1,1} = p$), so if $\Gamma_3 = [1, n-s+1] \times [1, m-s+1] - \{(1,1)\}$, then

$$p_{n,m,s} = nmpq^{n+m-2} p_{n-1,m-1,s-1} + \mathrm{err}_{n,m,s}, \quad (49)$$

where

$$\mathrm{err}_{n,m,s} \leqslant \sum_{(i,j)\in\Gamma_3} \binom{n}{i}\binom{m}{j} q^{i(m-j)+j(n-i)} p_{n-i,m-j,s-1}. \quad (50)$$

Note, that for $\beta_{n,m,s}$ we have

$$\beta_{n,m,s} = nmpq^{n+m-2} \beta_{n-1,m-1,s-1}, \quad (51)$$

so (49) and (51) are similar recursions, but unfortunately we could not find good bounds on the error term $\mathrm{err}_{n,m,s}$, especially for large s, to get an asymptotic upper bound on $p_{n,m,s}$.

6 Related Work

The problem of space-efficient encoding of relational databases is related to some other recent research efforts.

oarieli@mta.ac.il

The information content of a relational database schema was considered in a the papers of A. Benczúr [3], [4] but his model was entirely different, based on Kolgomorov complexity. Grumbach and Vianu [8] used standard encodings of complex object database instances on Turing tapes for efficient query answering and schema recovery. The size of their standard code of a database domain divided by the cardinality of the database domain has an upper bound of a polylogarithmic function of the cardinality of the database domain. Grumbach and Mecca [7] considered the problem of rediscovering the schema of nested relations that have been encoded as strings for storage purposes. Arenas and Libkin [2] has introduced a new information theoretical concept of relative information content of a position in the database and used it to justify Boyce-Codd normal forms. Kolahi and Libkin has successfully applied the concept for an information theoretic study on 3NF [9], XML design [10], and worst-case redundancy analysis [11]. Köhler[12] proposed and analyzed a new normal form for relational databases based on the idea of minimizing overall storage space and update costs.

7 Open Problems and Future Work

We initialized a study on the information content (smallest storage space) of databases of a given dependency schema. Partial results were proved for some simple sets of functional or multivalued dependencies.

Problem 1. Determine $\mathrm{Inf}(R, \Sigma)$ for other small sets of dependencies (fd's, mvd's, etc.) and analyze the coding problem for elementary modifications of the instance.

In section 2, we considered the case where the dependency schema consists of a single key. This gave rise to the concept of 2-distance preserving and strongly 2-distance preserving codes which were discussed in section 2 and 4. We proved a considerably weaker bound (Theorem 1) than (8), but it is at least exponential in b. We believe that the trivial bound is the best possible.

Conjecture 1. Let $2 \leqslant a, b$. If c is a 2-distance-preserving code of $(R, \{K \to \Omega\})$ then $|c| \geqslant 2^a(1 + 2^b)$.

(Strongly) d-distance preserving codes can be defined analogously by changing Hamming distance 2 to d in the definitions.

Problem 2. Give good bounds on the length of (strongly) d-distance preserving codes.

We also discussed the problem of determining $\mathrm{Inf}(R, \Sigma)$ for multivalued dependencies. We considered the example $\Omega = \{1, 2, 3\}$ and $\Sigma = \{\{1\} \twoheadrightarrow \{2\}, \{2\} \twoheadrightarrow \{1\}\}$ in section 3 and 5. This led to the problem of counting $c_{n,m,s}$, the number of bipartite graphs with partition sizes n, m and exactly s connected components that contain at least one edge.

The asymptotic bounds for small s's in section 5 support that the following may be true for all s.

oarieli@mta.ac.il

Conjecture 2.

$$c_{n,m,s} \approx \begin{cases} 1 - \beta_{n,m,s} & \text{if } s = 1 \\ \beta_{n,m,s} & \text{if } s > 1 \end{cases}, \tag{52}$$

where

$$\beta_{n,m,s} = \frac{n!}{(n-s)!}\frac{m!}{(m-s)!}p^{s-1}q^{(s-1)(m+n)-s(s-1)} \tag{53}$$

and $p = q = 1/2$.

If Conjecture 2 holds, then we have by substituting (52) into (20)

$$|\mathcal{I}(R,\Sigma)| \approx 1 + 2^{nm}(h_1 + h_2), \tag{54}$$

where

$$h_1 = (1 - \frac{nm}{2^{n+m-1}})2^k \tag{55}$$

and

$$h_2 = \sum_{s=2}^{\min\{n,m\}} \frac{n!}{(n-s)!}\frac{m!}{(m-s)!}\frac{1}{2^{s-1}}\frac{1}{2^{(s-1)(m+n)-s(s-1)}}(2^k-1)^s. \tag{56}$$

So we have

$$\log_2(\beta_{n,m,s}(2^k-1)^s) \approx s(k - n - m + 1 + \log_2(n-s) + \log_2(m-s) + s). \tag{57}$$

As an example, if $n \leqslant m$ and $k < m - 3 - \log_2(nm)$ then h_2 can be upper estimated by 1, so the information content is close to the lower bound of (21). On the other hand if $k > n + m$, h_2 becomes the significant term and the information content is far from the lower bound. As an extreme case, if $n = m$ and $k \gg n$, only the last term itself gives $|\mathcal{I}(R,\Sigma)| > (n!)^2 2^{n(k-n)}$, which implies a less than $2n^2$ difference in $\mathrm{Inf}(R,\Sigma)$ to the upper bound of (21), so it can be close in magnitude.

We also discussed the problem of updates in section 3. The problem behaves badly for tuple insertion. An example was discussed where insertion of a single tuple implies plenty of new tuples and breaks the structure of the instance. So the answer for the following problem might be negative.

Problem 3. Give a coding for (R,Σ), that has the property that small changes in the instances imply a small change in the code.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1994)
2. Arenas, M., Libkin, L.: An information-theoretic approach to normal forms for relational and XML data. Journal of the ACM 52, 246–283 (2005)

oarieli@mta.ac.il

3. Benczúr, A.: Information Measurement in Relational Data Bases. In: Biskup, J., Demetrovics, J., Paredaens, J., Thalheim, B. (eds.) MFDBS 1987. LNCS, vol. 305, pp. 1–9. Springer, Heidelberg (1988)
4. Benczúr, A.: The Evolution of Human Communication and the Information Revolution- A Mathematical Perspective. Mathematical and Computer Modelling 38(7-9), 691–708 (2003)
5. Bonferroni, C.E.: Teoria statistica delle classi e calcolo delle probabilità. Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze 8, 3–62 (1936)
6. Gilbert, E.N.: Random graphs. Annals of Mathematical Statistics 30(4), 1141–1144 (1959)
7. Grumbach, S., Mecca, G.: In Search of the Lost Schema. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 314–331. Springer, Heidelberg (1998)
8. Grumbach, S., Vianu, V.: Tractable query languages for complex object databases. Journal of Computer and System Sciences 51(2), 149–167 (1995)
9. Kolahi, S., Libkin, L.: On redundancy vs dependency preservation in normalization: an information-theoretic study of 3NF. In: Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2006, Chicago, IL, USA, pp. 114–123 (2006)
10. Kolahi, S., Libkin, L.: Dependency-preserving normalization of relational and XML data. Journal of Computer and System Sciences 73(4), 636–647 (2007)
11. Kolahi, S., Libkin, L.: An information-theoretic analysis of worst-case redundancy in database design. ACM Transactions on Database Systems 35(1), 1–32 (2010)
12. Köhler, H.: Global Database Design based on Storage Space and Update Time Minimization. Journal of Universal Computer Science 15(1), 195–240 (2009)

oarieli@mta.ac.il

On Lifted Inference for a Relational Probabilistic Conditional Logic with Maximum Entropy Semantics*

Annika Krämer and Christoph Beierle

Fak. für Mathematik und Informatik, FernUniversität in Hagen, 58084 Hagen, Germany

Abstract. When extending probabilistic logic to a relational setting, it is desirable to still be able to use efficient inference mechanisms developed for the propositional case. In this paper, we investigate the relational probabilistic conditional logic FO-PCL whose semantics employs the principle of maximum entropy. While in general, this semantics is defined via the ground instances of the rules in an FO-PCL knowledge base $\mathcal{R}$, the maximum entropy model can be computed on the level of rules rather than on the level of instances of the rules if $\mathcal{R}$ is parametrically uniform, thus providing lifted inference. We elaborate in detail the reasons precluding $\mathcal{R}$ from being parametrically uniform. Based on this investigation, we derive a new syntactic criterion for parametric uniformity and develop an algorithm that transforms any FO-PCL knowledge base $\mathcal{R}$ into an equivalent knowledge base $\mathcal{R}'$ that is parametrically uniform.

1 Introduction

One of the major challenges in representing real world phenomena in a knowledge based system is the modelling of uncertainty. There are different approaches addressing this issue by combining logic with probabilities, see e.g. the foundational work reported in [15,6,5]. Probabilistic conditional logic (PCL) [20] considers conditional probabilities $P(B|A)$ for conditionals of the form *if A then B* [1,16], formally denoted by $(B|A)$ (or just by (B) if A is a tautology). While these approaches are based upon propositional logic, more recently there has been a series of developments aiming at exploiting the expressiveness of relational or general first order logic in combination with probabilities, e.g. Bayesian logic programs and Markov logic networks (see [8] for an overview). A relational extension of PCL is the logic FO-PCL developed in [7] for which the following example illustrates both the expression of probabilistic relational statements and of specific knowledge about individuals.

Example 1 (Elephant keeper). The elephant keeper example, adapted from [4] and also [7], models the relationships among elephants in a zoo and their keepers. Elephants usually like their keepers, except for keeper Fred. But elephant Clyde gets along with everyone, and therefore he also likes Fred:

$$EK_1 : \langle (likes(E,K))[0.9], K \neq fred \rangle \qquad EK_3 : \langle (likes(clyde, fred))[1.0], \top \rangle$$
$$EK_2 : \langle (likes(E, fred))[0.05], E \neq clyde \rangle$$

* The research reported here was partially supported by the DFG - Deutsche Forschungsgemeinschaft (grant BE 1700/7-2).

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 224–243, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

EK_1 and EK_2 represent the statements that elephants normally like their keeper, except for Fred. EK_3 represents the exceptional fact that elephant Clyde likes Fred. The so-called constraint formulas $K \neq fred$ and $E \neq clyde$ of EK_1 and EK_2 make sure that there will not be an inconsistency when substituting the variables with constants. In EK_2, for example, the constraint formula prohibits the substitution of E by $clyde$. If the constraint formula of EK_2 was $\top$ instead (denoting a tautology), and if E was substituted by $clyde$, the result would be the ground conditional $\langle(likes(clyde, fred))[0.05], \top\rangle$, which would be inconsistent with EK_3 since no probability distribution can satisfy $likes(clyde, fred)$ both with probability 1.0 and with probability 0.05.

Inferencing in relational probabilistic logic has to take into account the variables occurring in the sentences of a knowledge base. In order to avoid inferencing on the level of all possible ground instances, lifted inference reasons at the level of quantified sentences. The aim of this work is to present an approach to lifted inference in FO-PCL.

The semantics of FO-PCL is based on the selection of a model of a knowledge base $\mathcal{R}$ that satisfies all conditionals in $\mathcal{R}$ and that is as unbiased as possible; this model is uniquely determined by the principle of maximum entropy [17,18,11,12]. Whereas in a propositional framework, there can be no instantiation of knowledge base elements, the maximum entropy model computation in FO-PCL requires to consider all ground instances of each conditional $R \in \mathcal{R}$. In order to ease this computationally expensive task, [7] develops the criterion of *parametric uniformity* for $\mathcal{R}$ such that lifted inference can be used if $\mathcal{R}$ fulfills this criterion.

The work in [10] defines notions of FO-PCL inter-rule and intra-rule interactions that preclude parametric uniformity for $\mathcal{R}$ and develops rules transforming $\mathcal{R}$ into $\mathcal{R}'$ not containing such interactions. However, while this approach removes all FO-PCL interactions as defined in [10], we will show here that it is incomplete in the sense that there are other reasons for parametric non-uniformity. For instance, we will show that $\mathcal{R}$ can fail to be parametrically uniform while being fully generic in the sense that no individual constant is mentioned in the conditionals in $\mathcal{R}$.

After briefly recalling the basic notions of FO-PCL (Sec. 2), we elaborate in detail the reasons for parametric non-uniformity and present the new and simpler criterion of *imbalanced sharing* and *imbalanced use*. (Sec. 3). Using knowledge bases in constraint normal form (Sec. 4), this enables us to precisely define interactions of conditionals as reasons for parametric non-uniformity (Sec. 5 and 6). The interaction removing transformation rules $\mathcal{PU}$ presented in Sec. 7 are correct and complete in the sense that $\mathcal{PU}(\mathcal{R})$ has the same maximum entropy model as $\mathcal{R}$ and is parametrically uniform. In Sec. 8, we discuss some further related work and conclude. Due to lack of space, we only give proof sketches in this paper; full technical proofs for all theorems are given in [14].

2 Overview of FO-PCL: Syntax and Semantics

FO-PCL uses function-free signatures of the form $\Sigma = (S, D, Pred)$ where S is a set of sorts, $D = \bigcup_{s \in S} D^{(s)}$ is a finite set of (disjoint) sets of sorted constant symbols, and $Pred$ is a set of predicate symbols, each coming with an arity of the form $s_1 \times \ldots \times s_n \in$

oarieli@mta.ac.il

S^n indicating the required sorts for the arguments. Variables $\mathcal{V}$ also have a unique sort, and all formulas and variable substitutions must obey the obvious sort restrictions. In the following, we will adopt the unique names assumption, that is, we require that different constants denote different elements.

An FO-PCL *conditional* $R = \langle(\phi_R|\psi_R)[\xi_R], C_R\rangle$ is composed of a *premise* ψ_R and a *conclusion* ϕ_R, which are quantifier and function free first-order formulas (over Σ and $\mathcal{V}$) without equality, a probability value $\xi_R \in [0,1]$, and a *constraint formula* C_R which is a quantifier-free first-order formula using only the equality predicate. For $\neg(V = X)$ we also write $(V \neq X)$, and $\top$ resp. $\bot$ denote a tautology resp. a contradiction. An *FO-PCL knowledge base* is a pair $(\Sigma, \mathcal{R})$ where $\mathcal{R}$ is a set of conditionals over $\Sigma, \mathcal{V}$. In the following, we will call just $\mathcal{R}$ a knowledge base and Σ will be given by the context. We will use the notation $C_R \vDash V \neq c$ to express that under the unique names assumption, the constraint formula C_R of a conditional R entails the constraint $V \neq c$, and $C_R \nvDash V \neq c$ means that C_R does not entail the constraint.

The constraint formula makes it possible to explicitly express that a generic conditional is not applicable with respect to a particular individual. Without constraint formulas, having a generic conditional and a corresponding conditional for a specific individual, that specific conditional might formally contradict the general one when considering all instances (cf. Example 1).

When the constraint formula of a ground instance of R evaluates to $true$, that instance is called *admissible*, and $gnd(R)$ denotes the set of all admissible instances of R (over Σ), in the following also just called instances. The Herbrand base $\mathcal{H}(\mathcal{R})$ is the set of all atoms in all $gnd(R)$ with $R \in \mathcal{R}$, and every subset of $\mathcal{H}(\mathcal{R})$ is a Herbrand interpretation, defining a logical semantics for $\mathcal{R}$. The probabilistic semantics of $\mathcal{R}$ is a possible world semantics [9] where the ground atoms in $\mathcal{H}(\mathcal{R})$ are binary random variables. An FO-PCL *interpretation* $p_{X(\mathcal{R})}$ of $\mathcal{R}$ is thus a joint probability function over $\mathcal{H}(\mathcal{R})$, and $p_{X(\mathcal{R})}$ is a *model* of $\mathcal{R}$ if it satisfies every $R \in \mathcal{R}$, where $p_{X(\mathcal{R})}$ satisfies R iff for every admissible instance $\langle(\phi|\psi)[\xi_R], \top\rangle$ of R it holds that $p_{X(\mathcal{R})}(\phi \wedge \psi) = \xi_R \cdot p_{X(\mathcal{R})}(\psi)$. Note that for the case of $P(\theta(\psi)) > 0$, this equation is equivalent to $\frac{P(\theta(\phi)\wedge\theta(\psi))}{P(\theta(\psi))} = \xi$ and thus to $P(\theta(\phi)|\theta(\psi)) = \xi$, expressing conditional probability.

A knowledge base $\mathcal{R} = \{R_1, \ldots, R_m\}$ may have many different models, and the principle of maximum entropy [17] provides a method to select a model that is optimal in the sense that it is the most unbiased one. The computation of the uniquely determined maximum entropy model $p^*_{X(\mathcal{R})}$ is an optimization problem that can be represented by a Gibbs distribution:

$$p^*_{X(\mathcal{R})}(x) = \frac{1}{Z} \exp\left(\sum_{k=1}^{m} \sum_{g_{R_k} \in gnd(R_k)} \lambda_{g_{R_k}} f_{g_{R_k}}(x)\right) \tag{1}$$

where $f_{g_{R_k}}$ is the feature function determined by g_{R_k}, $\lambda_{g_{R_k}}$ is a Lagrange multiplier and Z is a normalization constant. We will not elaborate on the details of Equation (1) as they are not important for the rest of this work (see [7] for a detailed explanation). What is important to note is that according to (1), one optimization parameter $\lambda_{g_{R_k}}$ has to be determined for *each single ground instance* g_{R_k} of each conditional R_k. This

oarieli@mta.ac.il

readily yields a computationally infeasible optimization problem for larger knowledge bases because there might be just too many ground instances.

However, there are FO-PCL knowledge bases for which the ground instances of a conditional share the same entropy-optimal parameter. Parametric uniformity [7] means that for all conditionals all their ground instances share the same entropy-optimal parameter value. The advantage of parametric uniformity is that just *one* optimization parameter $\lambda^*_{R_k}$ per conditional R_k has to be computed instead of one parameter per ground instance, yielding a usually computationally feasible problem:

$$p^*_{X(\mathcal{R})}(x) = \frac{1}{Z} \exp\left(\sum_{k=1}^{m} \lambda^*_{R_k} \sum_{g_{R_k} \in gnd(R_k)} f_{g_{R_k}}(x) \right) \tag{2}$$

Whereas parametric uniformity is a semantic notion, in [7] a syntactic criterion using so-called *probabilistic constraint involutions* sufficient to ensure it is presented. This syntactic criterion is based on the observation that parametric uniformity indicates identical knowledge about all ground instances of the same conditional for an FO-PCL knowledge base $\mathcal{R}$. Due to this, one should be able to transpose two ground instances g_{R_k}, g'_{R_k} of a conditional in $\mathcal{R}$ without changing the joint probability function with maximum entropy. In this case the transposed ground instances must possess the same entropy optimal parameter, as the Gibbs distribution in (1) is determined by a unique set of Lagrange multipliers. A *probabilistic constraint involution* transposes instances of conditionals with an involution $\pi_{F(\mathcal{R})}$ and it transposes ground atoms with an involution $\pi_{X(\mathcal{R})}$. An *involution covering* for $\mathcal{R}$ is a set

$$\Pi := \left\{ ({\pi_{F(\mathcal{R})}}^{(1)}, {\pi_{X(\mathcal{R})}}^{(1)}), \ldots, ({\pi_{F(\mathcal{R})}}^{(|\Pi|)}, {\pi_{X(\mathcal{R})}}^{(|\Pi|)}) \right\}$$

of pairs of $\pi_{F(\mathcal{R})}$ and $\pi_{X(\mathcal{R})}$, so that for any two instances $g_{R_k}, g'_{R_k} \in gnd(R_k)$ with $R_k \in \mathcal{R}$, there exists a sequence $\langle i_1, \ldots, i_n \rangle$ of indices $i_j \in \{1, \ldots, |\Pi|\}$, such that

$${\pi_{F(\mathcal{R})}}^{(i_j)}(g_k^{i_{j-1}}) = g_k^{i_j}, 1 \leqslant j \leqslant n$$

holds, with $g_k^{i_j} \in gnd(R_k)$ for all $1 \leqslant j \leqslant n$, $g_k^{i_0} := g_{R_k}$ and $g_k^{i_n} := g'_{R_k}$ (Def. 7.4.2 in [7]).

Example 2. We continue Example 1 where *likes* takes one argument of sort *Elephant* and one of sort *Keeper*. If D contains the constants $\{clyde, dumbo, nirvan\}$ of sort *Elephant* and the constants $\{fred, paul, simon\}$ of sort *Keeper*, the set of ground instances of the conditionals is:

$gr_{EK_{1-1}}$: $\langle (likes(clyde, paul))[0.9], \top \rangle$
$gr_{EK_{1-2}}$: $\langle (likes(dumbo, paul))[0.9], \top \rangle$
$gr_{EK_{1-3}}$: $\langle (likes(nirvan, paul))[0.9], \top \rangle$
$gr_{EK_{1-4}}$: $\langle (likes(clyde, simon))[0.9], \top \rangle$
$gr_{EK_{1-5}}$: $\langle (likes(dumbo, simon))[0.9], \top \rangle$
$gr_{EK_{1-6}}$: $\langle (likes(nirvan, simon))[0.9], \top \rangle$
$gr_{EK_{2-1}}$: $\langle (likes(dumbo, fred))[0.05], \top \rangle$
$gr_{EK_{2-2}}$: $\langle (likes(nirvan, fred))[0.05], \top \rangle$
$gr_{EK_{3-1}}$: $\langle (likes(clyde, fred))[1.0], \top \rangle$

A possible probabilistic constraint involution $\Pi^{(1)}$ is:

oarieli@mta.ac.il

$\pi^{(1)}_{F(\mathcal{R})} = (gr_{EK_{1-1}} \quad gr_{EK_{1-2}}) \quad \pi^{(1)}_{X(\mathcal{R})} = (likes(clyde, paul) \quad likes(dumbo, paul))$

Another probabilistic constraint involution $\Pi^{(2)}$ would be:

$\pi^{(2)}_{F(\mathcal{R})} = (gr_{EK_{1-1}} \quad gr_{EK_{1-4}}) \quad \pi^{(2)}_{X(\mathcal{R})} = (likes(clyde, paul) \quad likes(clyde, simon))$

For Example 2, it is particularly easy to find an involution covering as each ground instance of a conditional uses only one ground atom, and each ground atom appears in only one ground instance. Later on, we will present more complex examples and situations, where for a given knowledge base no involution covering exists. The importance of involution coverings is given by Corollary 7.4.4 in [7] which states:

Theorem 1 (Involution covering implies parametric uniformity). *If there is an involution covering for $\mathcal{R}$, then $\mathcal{R}$ is parametrically uniform.*

3 Imbalanced Sharing and Imbalanced Use of Ground Atoms

In this section, we will elaborate the reasons precluding the existence of an involution covering in detail. This elaboration yields a syntactic criterion for parametric non-uniformity that is simpler than the rather complex notion of involution covering: The imbalanced sharing or use of ground atoms or of combinations of ground atoms.

Example 3 (Imbalanced sharing). The misanthrope example, adapted from [7], models friendship relations between a group of people, with one exceptional member, a misanthrope. In general, if a person V likes another person U, then it is very likely that U likes V, too. But there is one person, the misanthrope, who generally does not like other people:

$MI_1 : \langle (likes(U, V) | likes(V, U))[0.9], U \neq V \rangle$
$MI_2 : \langle (likes(a, V))[0.05], V \neq a \rangle$

The set of ground instances for the set of constant symbols $D = \{a, b, c\}$ is:

$gr_{MI_{1-1}}$: $\langle (likes(a, b) | likes(b, a))[0.9], \top \rangle$ $\quad gr_{MI_{1-5}}$: $\langle (likes(c, a) | likes(a, c))[0.9], \top \rangle$
$gr_{MI_{1-2}}$: $\langle (likes(a, c) | likes(c, a))[0.9], \top \rangle$ $\quad gr_{MI_{1-6}}$: $\langle (likes(c, b) | likes(b, c))[0.9], \top \rangle$
$gr_{MI_{1-3}}$: $\langle (likes(b, a) | likes(a, b))[0.9], \top \rangle$ $\quad gr_{MI_{2-1}}$: $\langle (likes(a, b))[0.05], \top \rangle$
$gr_{MI_{1-4}}$: $\langle (likes(b, c) | likes(c, b))[0.9], \top \rangle$ $\quad gr_{MI_{2-2}}$: $\langle (likes(a, c))[0.05], \top \rangle$

One probabilistic constraint involution $\Pi^{(1)}$ for this example would be:

$\pi^{(1)}_{F(\mathcal{R})} = (gr_{MI_{1-4}} \quad gr_{MI_{1-6}}) \qquad \pi^{(1)}_{X(\mathcal{R})} = (likes(b, c) \quad likes(c, b))$

Another probabilistic constraint involution $\Pi^{(2)}$ would be:

$$\pi^{(2)}_{F(\mathcal{R})} = \begin{matrix} (gr_{MI_{1-1}} & gr_{MI_{1-2}}) \\ (gr_{MI_{2-1}} & gr_{MI_{2-2}}) \end{matrix} \qquad \pi^{(2)}_{X(\mathcal{R})} = \begin{matrix} (likes(a, b) & likes(a, c)) \\ (likes(b, a) & likes(c, a)) \end{matrix}$$

For constructing $\Pi^{(2)}$, we start with the two ground conditionals $gr_{MI_{1-1}}$ and $gr_{MI_{1-2}}$. Adding their transposition to $\pi^{(2)}_{F(\mathcal{R})}$ enforces adding the transpositions $(likes(a, b) \quad likes(a, c))$ and $(likes(b, a) \quad likes(c, a))$ to $\pi^{(2)}_{X(\mathcal{R})}$. As the ground atoms $(likes(a, b))$ and $likes(a, c))$ are also used in $gr_{MI_{2-1}}$ and $gr_{MI_{2-2}}$, this leads to the necessity of adding the transposition $(gr_{MI_{2-1}} \quad gr_{MI_{2-2}})$ to $\pi^{(2)}_{F(\mathcal{R})}$.

oarieli@mta.ac.il

However, if we try to construct a probabilistic constraint involution $\Pi^{(3)}$ starting with the two ground instances $gr_{MI_{1-1}}$ and $gr_{MI_{1-3}}$, we encounter a problem: Adding the transposition $(gr_{MI_{1-1}} \quad gr_{MI_{1-3}})$ to $\pi^{(3)}_{F(\mathcal{R})}$ requires adding the transposition $(likes(a,b) \quad likes(b,a))$ to $\pi^{(3)}_{X(\mathcal{R})}$. As the ground atom $likes(a,b)$ is also used in $gr_{MI_{2-1}}$, we would have to transpose $gr_{MI_{2-1}}$ as well, but there is no corresponding ground instance $\langle(likes(b,a))[0.05], \top\rangle$ to transpose it with. Thus, there is no probabilistic constraint involution that transposes $gr_{MI_{1-1}}$ with $gr_{MI_{1-3}}$, the problem being that ground instances of *likes* are shared by MI_1 and MI_2 in an imbalanced way.

Throughout the rest of this work, we will always assume that $\mathcal{R}$ denotes a knowledge base over $\Sigma = (S, D, Pred)$ and that $\mathcal{R}$ is *consistent* (i.e., it has at least one model). With $const(\mathcal{R}) \subseteq D$ we denote the set of all constants occurring in the conditionals of $\mathcal{R}$. Furthermore, for a conditional $R \in \mathcal{R}$ we will use the following notations:

- $at(R)$ denotes the set of all atoms occurring in R.
- $GA_{P_R} = \{p \mid \text{there is } g_R \in gnd(R) \text{ with } p \in at(g_R)\}$ denotes the set of ground atoms of P_R occurring in the groundings $gnd(R)$, for any atom $P_R \in at(R)$.
- $GA_{\langle P_R, Q_R\rangle} = \{\langle p, q\rangle \mid \text{there is } g_R \in gnd(R) \text{ with } p,\, q \in at(g_R)\}$ denotes the set of combinations (tuples) of ground atoms of P_R and Q_R occurring in $gnd(R)$, for any two different atoms $P_R,\, Q_R \in at(R)$.
- $|p|_{gnd(R)} = |\{g_R \mid g_R \in gnd(R) \text{ and } p \in at(g_R)\}|$ denotes the number of ground instances of R in which p occurs, where p is a ground instance of an atom $P_R \in at(R)$.
- $|\langle p, q\rangle|_{gnd(R)} = |\{g_R \mid g_R \in gnd(R) \text{ and } p, q \in at(g_R)\}|$ denotes the number of ground instances of R in which both p and q occur, where p is a ground instance of an atom $P_R \in at(R)$ and q is a ground instance of a different atom $Q_R \in at(R)$.

Definition 1 (imbalanced sharing). *Let $R_1,\, R_2 \in \mathcal{R}$ be two different conditionals, and let $P_{R_1} \in at(R_1)$ and $P_{R_2} \in at(R_2)$.*

1. *There is an* imbalanced sharing of ground atoms *by R_1 and R_2 (with respect to P_{R_1} and P_{R_2}) iff $GA_{P_{R_1}} \cap GA_{P_{R_2}} \neq \emptyset$ and $GA_{P_{R_1}} \neq GA_{P_{R_2}}$.*
2. *Let $Q_{R_1} \in at(R_1)$ and $Q_{R_2} \in at(R_2)$ be atoms different from P_{R_1} and P_{R_2}, respectively. There is an* imbalanced sharing of combinations of ground atoms *by R_1 and R_2 (with respect to P_{R_1}, Q_{R_1} and P_{R_2}, Q_{R_2}) iff $GA_{\langle P_{R_1}, Q_{R_1}\rangle} \cap GA_{\langle P_{R_2}, Q_{R_2}\rangle} \neq \emptyset$ and $GA_{\langle P_{R_1}, Q_{R_1}\rangle} \neq GA_{\langle P_{R_2}, Q_{R_2}\rangle}$.*
3. *There is an* imbalanced sharing in $\mathcal{R}$ *iff there are conditionals in $\mathcal{R}$ such that (1.) or (2.) is the case.*

An imbalanced sharing of ground atoms means that the ground instances of R_1 and R_2 use at least one identical ground atom of P_{R_1} and P_{R_2}, but the sets of ground atoms of P_{R_1} and P_{R_2} are not equal. An imbalanced sharing of combinations of ground atoms means that the ground instances of R_1 and R_2 use at least one identical combination of ground atoms of P_{R_1} and Q_{R_1}, respectively P_{R_2} and Q_{R_2}, but the sets of combinations of ground atoms of P_{R_1} and Q_{R_1}, respectively P_{R_2} and Q_{R_2}, are not equal.

The notion of imbalanced sharing of combinations of ground atoms can be extended to n-tuples given by $GA_{\langle P^1_{R_i}, \ldots, P^n_{R_i}\rangle} = \{\langle p^1, \ldots, p^n\rangle \mid \text{there is } g_{R_i} \in$

oarieli@mta.ac.il

$gnd(R_i)$ with $p^1 \in at(g_{R_i}), \ldots, p^n \in at(g_{R_i})\}$ for a conditional R_i and n different atoms $P^1_{R_i}, \ldots, P^n_{R_i} \in at(R_i)$. However, any imbalanced sharing of n-tuples of ground atoms can be reduced to one or several cases of an imbalanced sharing of 2-tuples of ground atoms:

Proposition 1 (imbalanced sharing of n-tuples). *Let $R_1, R_2 \in \mathcal{R}$ be two different conditionals. If there are atoms $P^1_{R_1}, \ldots, P^n_{R_1} \in at(R_1)$ and $P^1_{R_2}, \ldots, P^n_{R_2} \in at(R_2)$ such that $GA_{\langle P^1_{R_1}, \ldots, P^n_{R_1}\rangle} \cap GA_{\langle P^1_{R_2}, \ldots, P^n_{R_2}\rangle} \neq \emptyset$ and $GA_{\langle P^1_{R_1}, \ldots, P^n_{R_1}\rangle} \neq GA_{\langle P^1_{R_2}, \ldots, P^n_{R_2}\rangle}$, then there are atoms $P^i_{R_1}, P^j_{R_1} \in at(R_1)$ and $P^i_{R_2}, P^j_{R_2} \in at(R_2)$ such that $GA_{\langle P^i_{R_1}, P^j_{R_1}\rangle} \cap GA_{\langle P^i_{R_2}, P^j_{R_2}\rangle} \neq \emptyset$ and $GA_{\langle P^i_{R_1}, P^j_{R_1}\rangle} \neq GA_{\langle P^i_{R_2}, P^j_{R_2}\rangle}$.*

While an imbanced sharing of ground atoms refers to two different conditionals, an imbalanced use of ground atoms may occur within a single conditional R.

Example 4 (Imbalanced use). The ground instances of the conditional $R = \langle (P(U,V)|Q(V))[\xi], U \neq V \wedge U \neq a\rangle$ for the set of constants $D = \{a, b, c\}$ are:

$$gr_{R-1} = \langle (P(b,a)|Q(a))[\xi], \top\rangle \qquad gr_{R-3} = \langle (P(c,a)|Q(a))[\xi], \top\rangle$$
$$gr_{R-2} = \langle (P(b,c)|Q(c))[\xi], \top\rangle \qquad gr_{R-4} = \langle (P(c,b)|Q(b))[\xi], \top\rangle$$

As can be seen, the ground atom $Q(a)$ occurs twice, while the ground atoms $Q(b)$ and $Q(c)$ occur only once. If we try to construct a probabilistic constraint involution starting with the transposition $(gr_{R-1} \quad gr_{R-2})$ in $\pi_{F(R)}$, we have to add the transposition $(Q(a) \quad Q(c))$ to $\pi_{X(R)}$. As $Q(a)$ is also used by gr_{R-3}, we would have to find another ground instance of R using $Q(c)$ to transpose gr_{R-3} with, but there is no such instance.

In general, an imbalanced use of ground atoms means that a particular ground atom of an atom P_R appears more often in the ground instances of R than another ground atom of P_R. An imbalanced use of combinations of ground atoms means that a particular combination of ground atoms of two atoms P_R and Q_R appears more often in the ground instances of R than another combination of ground atoms of P_R and Q_R.

Definition 2 (imbalanced use). *Let $R \in \mathcal{R}$ and $P_R \in at(R)$.*

1. *There is an* imbalanced use of ground atoms *by R (with respect to P_R) iff there are $p, p' \in GA_{P_R}$ such that $|p|_{GI_R} < |p'|_{GI_R}$.*
2. *Let $Q_R \in at(R)$ be different from P_R. There is an* imbalanced use of combinations of ground atoms *by R (with respect to P_R and Q_R) iff there are $p, p' \in GA_{P_R}$ and $q, q' \in GA_{Q_R}$ such that $|\langle p, q\rangle|_{GI_R} < |\langle p', q'\rangle|_{GI_R}$.*
3. *There is an* imbalanced use in $\mathcal{R}$ *iff there is a conditional in $\mathcal{R}$ such that (1.) or (2.) is the case.*

Note that the two atoms P_R and Q_R might use the same predicate symbol.

Definition 3 (balanced and imbalanced knowledge base). *$\mathcal{R}$ is* imbalanced *iff there is an imbalanced sharing or an imbalanced use in $\mathcal{R}$, otherwise $\mathcal{R}$ is* balanced.

This gives us a syntactic criterion for the existence of an involution covering and thus for parametric uniformity of a knowledge base $\mathcal{R}$. If there is an imbalanced sharing or use in $\mathcal{R}$, then no involution covering for $\mathcal{R}$ can be found. On the other hand, if there is no imbalanced sharing or use in $\mathcal{R}$, then the balanced property of $\mathcal{R}$ ensures that an involution covering for $\mathcal{R}$ exists.

oarieli@mta.ac.il

Theorem 2 (correctness and completeness of the balanced criterion). *Let $\mathcal{R}$ be a consistent knowledge base. There is an involution covering for $\mathcal{R}$ iff $\mathcal{R}$ is balanced.*

Both involution covering and the balanced criterion refer to the set of constants D given by the used signature $\Sigma = (S, D, Pred)$. It is possible that just due to the lack of enough constants there is no imbalance (and thus an involution covering), i.e. there is no imbalance w.r.t. D, but there is one w.r.t. D' with $D \subseteq D'$ [14]. For the remaining sections, we therefore assume that $\mathcal{R}$ is a knowledge base over $\Sigma = (S, D, Pred)$ containing enough constants such that the degenerated case of an imbalance just due to the lack of enough constants does not occur. For instance, in the one-sorted case where $|S| = 1$, Σ has enough constants if $|D| \geqslant |const(\mathcal{R})| + max_v$ where max_v denotes the maximal number of different variables occurring in a conditional $R \in \mathcal{R}$.

Theorem 3 (imbalance/balance not changed by more constants). *Let $\mathcal{R}$ be consistent. If $\mathcal{R}$ is imbalanced, then $\mathcal{R}$ is imbalanced w.r.t. any D' with $D \subseteq D'$. If $\mathcal{R}$ is balanced and Σ has enough constants, then $\mathcal{R}$ is balanced w.r.t. any D' with $D \subseteq D'$.*

4 Knowledge Bases in Constraint Normal Form

While in general, FO-PCL constraints may contain conjunctions and disjunctions, for the rest of the paper we will consider only FO-PCL conditionals with a constraint formula which is a conjunction of equations and inequations. This is not a principle restriction, since conditionals of the form $\langle(\phi|\psi)[\xi], C_1 \vee C_2\rangle$ may be replaced by the two conditionals $\langle(\phi|\psi)[\xi], C_1\rangle$ and $\langle(\phi|\psi)[\xi], C_2\rangle$.

When transforming a knowledge base $\mathcal{R}$, we assume that the original knowledge base $\mathcal{R}$ is consistent. This implies that each $R \in \mathcal{R}$ is *self-consistent* which means that there is at least one model for R (the term *self-consistent* is adopted from Chapter 2 of [21]). For example, a conditional like $\langle(Q(a) \wedge \neg Q(a))[0.7], \top\rangle$ is not self-consistent. Self-inconsistent conditionals may not be confused with conditionals that have an inconsistent constraint formula, like $X = c \wedge X \neq c$. As those conditionals have no ground instances, they are trivially self-consistent. We will call conditionals that have a consistent constraint formula *constraint-consistent* and those that have an inconsistent constraint-formula *constraint-inconsistent*. Additionally, it will be convenient to have all $R \in \mathcal{R}$ in *constraint normal form*.

Definition 4 (constraint normal form). *Let $R = \langle(\phi|\psi)[\xi], C\rangle$ be self-consistent.*

1. *If R is constraint-consistent and κ is a constraint occurring in C, then κ is* redundant *in C if one of the following conditions applies:*
   - *κ is an equation of a variable or constant with itself.*
   - *κ is an inequation of a constant with a different constant.*
   - *κ is an inequation of a variable with a constant and there is an equation of the same variable with a different constant.*
   - *κ is equivalent to another constraint κ' ($\kappa \equiv \kappa'$).*
   - *κ refers to a variable that does not occur in $\phi \cup \psi$.*
2. *R is in* constraint normal form *if R is constraint-consistent and C is a conjunction of negated non-redundant constraints, or R is constraint-inconsistent and $C = \bot$.*

oarieli@mta.ac.il

3. *A knowledge base $\mathcal{R}$ is in* constraint normal form *if every conditional $R \in \mathcal{R}$ is in constraint normal form.*

Proposition 2 (constraint normal form operator). *There is an operator ν that transforms any self-consistent conditional R into $\nu(R)$ such that R and $\nu(R)$ have the same ground instances and $\nu(R)$ is in constraint normal form.*

In the following, we will use such an operator ν which can easily be defined. For instance, ν removes all equations of the form $X = a$ from C and applies the corresponding substitution $\sigma = \{X/a\}$ to ϕ and ψ and the remaining part of C. The obvious extension of ν to $\mathcal{R}$ is denoted by $\nu(\mathcal{R})$ or just by $\nu\mathcal{R}$.

5 Imbalanced Sharing and Inter-Rule Interactions

We study in detail the reasons for imbalanced sharing and develop a syntactic criterion that is independent of the constants occurring in $D \setminus const(\mathcal{R})$.

5.1 Reasons for an Imbalanced Sharing of Ground Atoms

In Example 3, we have seen that an imbalanced sharing of ground atoms occurs if there is a conditional R_1, with a constant c occurring at position n of an atom P_{R_1}, using a predicate symbol P, while in another conditional R_2, a variable V occurs at position n of an atom P_{R_2}, which also uses the predicate symbol P, and the constraint formula of R_2 doesn't exclude c as an admissible substitution for V. The fact that the ground instances of R_1 and R_2 share the ground atoms $P(\ldots, c, \ldots)$, but the ground instances of R_2 also use ground atoms $P(\ldots, d, \ldots)$, using a different constant d at position n, that are not used by R_1, prevents an involution covering: When transposing a ground instance of R_2 using $P(\ldots, c, \ldots)$ with a ground instance using $P(\ldots, d, \ldots)$, this requires the transposition of the grounds atom $P(\ldots, c, \ldots)$ and $P(\ldots, d, \ldots)$. The ground instances of R_1 using $P(\ldots, c, \ldots)$ would then have to be transposed accordingly, but there are no ground instances of R_1 using $P(\ldots, d, \ldots)$ to transpose them with. The same happens if both P_{R_1} and P_{R_2} contain a variable at position n, and the constraint formula of R_1 contains a restriction for the substitution of the variable at position n of P_{R_1}, while the constraint formula of R_2 doesn't contain an equivalent restriction for the variable at position n of P_{R_2}. In both cases, $GA_{P_{R_1}}$ and $GA_{P_{R_2}}$ share at least one element, but $GA_{P_{R_2}}$ contains at least one element that is not contained in $GA_{P_{R_1}}$.

Example 5 (Imbalanced sharing of ground atoms 1). Let $R_1 = \langle (Q(U)|P(V))[0.9], V \neq a \rangle$ and $R_2 = \langle (P(X))[0.5], \top \rangle$. R_1 and R_2 both use an atom with the predicate symbol P. The ground instances for the set of constants $D = \{a, b, c\}$ are:

$$
\begin{array}{ll}
gr_{R_{1-1}} = \langle (Q(a)|P(b))[0.9], \top \rangle & gr_{R_{1-6}} = \langle (Q(c)|P(c))[0.9], \top \rangle \\
gr_{R_{1-2}} = \langle (Q(a)|P(c))[0.9], \top \rangle & \\
gr_{R_{1-3}} = \langle (Q(b)|P(b))[0.9], \top \rangle & gr_{R_{2-1}} = \langle (P(a))[0.5], \top \rangle \\
gr_{R_{1-4}} = \langle (Q(b)|P(c))[0.9], \top \rangle & gr_{R_{2-2}} = \langle (P(b))[0.5], \top \rangle \\
gr_{R_{1-5}} = \langle (Q(c)|P(b))[0.9], \top \rangle & gr_{R_{2-3}} = \langle (P(c))[0.5], \top \rangle
\end{array}
$$

oarieli@mta.ac.il

The ground instances of R_1 use only the ground atoms $P(b)$ and $P(c)$, while the ground instances of R_2 use those same ground atoms as well as the ground atom $P(a)$. If we start a probabilistic constraint involution with $gr_{R_{2-1}}$ and $gr_{R_{2-2}}$, we have to transpose the ground atoms $P(a)$ and $P(b)$. As $gr_{R_{1-1}}$ also uses $P(b)$, we need to transpose $gr_{R_{1-1}}$ with a ground instance of R_1 using P(a). But there is no such ground instance.

In order to make an involution covering possible in Example 5, we could transform the conditional R_2, such that we obtain one conditional R_{2-1} that does not share ground atoms of the atom with the predicate symbol P with R_1, and one conditional R_{2-2}, using exactly the same ground atoms of the atom with the predicate symbol P as R_1, i.e., $R_{2-1} = \langle (P(a))[0.5], \top \rangle$ and $R_{2-2} = \langle (P(X))[0.5], X \neq a \rangle$.

As we have seen, an imbalanced sharing of ground atoms happens if in R_1, the admissible constants at position n of P_{R_1} are somehow restricted, while R_2 doesn't contain such a restriction regarding the admissible constants at position n of P_{R_2}. Whereas so far, we have only considered restrictions with respect to a particular constant c, affecting just a single position n of P_{R_1}, an imbalanced sharing can also happen if the restriction in R_1 affects two variables occurring at positions n and m of the atom P_{R_1}, thus making those two positions of P_{R_1} interdependent, while this kind of interdependency doesn't exist in R_2. This is demonstrated in the following example; note that this example involves only conditionals that do not use any constants at all, i.e., $const(\mathcal{R}) = \emptyset$. This situation is not covered by the definition of an interaction in [10].

Example 6. (Imbalanced sharing of ground atoms 2) Let $R_1 = \langle (S(Y)|P(U,V))[\xi_1], Y \neq U \wedge Y \neq V \wedge U \neq V \rangle$ and $R_2 = \langle (P(W,X))[\xi_2], \top \rangle$. R_1 and R_2 both use an atom with the predicate symbol P. The ground instances of R_1 and R_2 for the set of constants $D = \{a, b, c\}$ are:

$$
\begin{array}{ll}
gr_{R_{1-1}} = \langle (S(a)|P(b,c))[\xi_1], \top \rangle & gr_{R_{2-2}} = \langle (P(a,b))[\xi_2], \top \rangle \\
gr_{R_{1-2}} = \langle (S(a)|P(c,b))[\xi_1], \top \rangle & gr_{R_{2-3}} = \langle (P(a,c))[\xi_2], \top \rangle \\
gr_{R_{1-3}} = \langle (S(b)|P(a,c))[\xi_1], \top \rangle & gr_{R_{2-4}} = \langle (P(b,a))[\xi_2], \top \rangle \\
gr_{R_{1-4}} = \langle (S(b)|P(c,a))[\xi_1], \top \rangle & gr_{R_{2-5}} = \langle (P(b,b))[\xi_2], \top \rangle \\
gr_{R_{1-5}} = \langle (S(c)|P(a,b))[\xi_1], \top \rangle & gr_{R_{2-6}} = \langle (P(b,c))[\xi_2], \top \rangle \\
gr_{R_{1-6}} = \langle (S(c)|P(b,a))[\xi_1], \top \rangle & gr_{R_{2-7}} = \langle (P(c,a))[\xi_2], \top \rangle \\
 & gr_{R_{2-8}} = \langle (P(c,b))[\xi_2], \top \rangle \\
gr_{R_{2-1}} = \langle (P(a,a))[\xi_2], \top \rangle & gr_{R_{2-9}} = \langle (P(c,c))[\xi_2], \top \rangle
\end{array}
$$

The ground instances of R_1 and R_2 use ground atoms with the predicate symbol P in an imbalanced way: The ground instances of R_1 only use those ground atoms in which the variables at position 1 and 2 are substituted by different constants, while the ground instances of R_2 use not only those ground atoms, but also those in which the variables at position 1 and 2 are substituted by the same constant. This imbalance prevents an involution covering: When transposing one of the ground instances of R_2 in which those two variables are substituted by different constants with one of the ground instances of R_2 in which those variables are substituted by the same constant, say, $gr_{R_{2-2}}$ using $P(a,b)$ with $gr_{R_{2-1}}$ using $P(a,a)$, then, for the ground instance $gr_{R_{1-5}}$, which also uses $P(a,b)$, there is no corresponding ground instance of R_1 using $P(a,a)$ to transpose it with.

oarieli@mta.ac.il

5.2 Reasons for an Imbalanced Sharing of Combinations of Ground Atoms

In Example 6, demonstrating the imbalanced sharing of ground atoms, there is a constraint that contains a restriction for two variables that both occur in an atom $P_{R_1} \in at(R_1)$. In the following example, demonstrating the imbalanced sharing of *combinations of ground atoms*, the two variables that the constraint refers to occur in two different atoms $P_{R_1} \in at(R_1)$ and $Q_{R_1} \in at(R_1)$.

Example 7 (Imbalanced sharing of combinations of ground atoms). Let $R_1 = \langle (S(U)|P(V))[\xi_1], U \neq V \rangle$ and $R_2 = \langle (P(X)|S(Y))[\xi_2], \top \rangle$. Both conditionals use an atom with the predicate symbol P and an atom with the predicate symbol Q. The ground instances of R_1 and R_2 for the set of constants $D = \{a, b, c\}$ are:

$$
\begin{array}{ll}
gr_{R_{1-1}} = \langle (S(a)|P(b))[\xi_1], \top \rangle & gr_{R_{2-2}} = \langle (P(a)|S(b))[\xi_2], \top \rangle \\
gr_{R_{1-2}} = \langle (S(a)|P(c))[\xi_1], \top \rangle & gr_{R_{2-3}} = \langle (P(a)|S(c))[\xi_2], \top \rangle \\
gr_{R_{1-3}} = \langle (S(b)|P(a))[\xi_1], \top \rangle & gr_{R_{2-4}} = \langle (P(b)|S(a))[\xi_2], \top \rangle \\
gr_{R_{1-4}} = \langle (S(b)|P(c))[\xi_1], \top \rangle & gr_{R_{2-5}} = \langle (P(b)|S(b))[\xi_2], \top \rangle \\
gr_{R_{1-5}} = \langle (S(c)|P(a))[\xi_1], \top \rangle & gr_{R_{2-6}} = \langle (P(b)|S(c))[\xi_2], \top \rangle \\
gr_{R_{1-6}} = \langle (S(c)|P(b))[\xi_1], \top \rangle & gr_{R_{2-7}} = \langle (P(c)|S(a))[\xi_2], \top \rangle \\
 & gr_{R_{2-8}} = \langle (P(c)|S(b))[\xi_2], \top \rangle \\
gr_{R_{2-1}} = \langle (P(a)|S(a))[\xi_2], \top \rangle & gr_{R_{2-9}} = \langle (P(c)|S(c))[\xi_2], \top \rangle
\end{array}
$$

The ground instances of R_1 and R_2 both use the ground atoms $P(a)$, $P(b)$, $P(c)$, $S(a)$, $S(b)$ and $S(c)$. But they use combinations of ground atoms with the predicate symbols P and S in an imbalanced way: The ground instances of R_1 only use combinations in which the parameters of the two atoms are substituted by different constants, while the ground instances of R_2 use not only those combinations, but also those in which the parameters are substituted by the same constant. This imbalance prevents an involution covering: When transposing one of the ground instances of R_2 in which those two variables are substituted by different constants with one of the ground instances of R_2 in which those variables are substituted by the same constant, say, $gr_{R_{2-2}}$ using the combination $\langle P(a), S(b) \rangle$ with $gr_{R_{2-1}}$ using the combination $\langle P(a), S(a) \rangle$, then for the ground instance $gr_{R_{1-3}}$, which also uses the combination $\langle P(a), S(b) \rangle$, there is no corresponding ground instance of R_1 using the combination $\langle P(a), S(a) \rangle$ to transpose it with.

5.3 Inter-Rule Interactions

Having analyzed what causes an imbalanced sharing of ground atoms or combinations of ground atoms, we can now define *inter-rule interactions* as constellations in which those imbalances occur. In order to simplify this definition, we will first define constellations in which the ground instances of R_1 and R_2 can impossibly share any ground atoms with respect to two atoms $P_{R_1} \in atoms(R_1)$ and $P_{R_2} \in atoms(R_2)$, thus no imbalance can occur.

Definition 5 (g-disjoint). *Let $R_1, R_2 \in \mathcal{R}$, and let $P_{R_1} \in at(R_1)$ and $P_{R_2} \in at(R_2)$. R_1 and R_2 are* g-disjoint with respect to P_{R_1} and P_{R_2} *if $GA_{P_{R_1}} \cap GA_{P_{R_2}} = \emptyset$.*

oarieli@mta.ac.il

Obviously, R_1 and R_2 are g-disjoint with respect to P_{R_1} and P_{R_2} if P_{R_1} and P_{R_2} use different predicate symbols. The following definition provides a syntactic criterion for g-disjointness although both P_{R_1} and P_{R_2} use the same predicate symbol.

Definition 6 (interaction-safe). *Let $R_1, R_2 \in \mathcal{R}$ with $R_1 = \langle(\phi_1|\psi_1)[\xi_1], C_1\rangle$ and $R_2 = \langle(\phi_2|\psi_2)[\xi_2], C_2\rangle$, and let $P_{R_1} \in at(R_1)$ and $P_{R_2} \in at(R_2)$ both use the predicate symbol P. Let U, V, W and Z be variables, $U \neq W$ and $V \neq Z$, and let c and d be different constants. R_1 and R_2 are* interaction-safe with respect to P_{R_1} and P_{R_2} *if one of the following conditions applies:*

(1) $P_{R_1} = P(\ldots, c, \ldots)$ *and* $P_{R_2} = P(\ldots, d, \ldots)$.
(2) $P_{R_1} = P(\ldots, c, \ldots)$ *and* $P_{R_2} = P(\ldots, V, \ldots)$ *and* $C_2 \models V \neq c$.
(3) $P_{R_1} = P(\ldots, U, \ldots, W, \ldots)$ *and* $C_1 \models U \neq W$ *and either*
- $P_{R_2} = P(\ldots, c, \ldots, c, \ldots)$, *or*
- $P_{R_2} = P(\ldots, V, \ldots, V, \ldots)$

(4) $P_{R_1} = P(\ldots, U, \ldots, U, \ldots)$ *and either*
- $P_{R_2} = P(\ldots, c, \ldots, d, \ldots)$, *or*
- $P_{R_2} = P(\ldots, V, \ldots, c, \ldots)$ *and* $C_2 \models V \neq c$, *or*
- $P_{R_2} = P(\ldots, c, \ldots, V, \ldots)$ *and* $C_2 \models V \neq c$, *or*
- $P_{R_2} = P(\ldots, V, \ldots, Z, \ldots)$ *and* $C_2 \models V \neq Z$.

Note that *interaction-safe* does not refer to any constants in $D \setminus const(\mathcal{R})$. Since any of the situations (1), (2), (3) or (4) in Def. 6 implies $GA_{P_{R_1}} \cap GA_{P_{R_2}} = \emptyset$, we have:

Proposition 3 (interaction-safe implies g-disjoint). *If R_1 and R_2 are interaction-safe with respect to P_{R_1} and P_{R_2}, then they are g-disjoint with respect to P_{R_1} and P_{R_2}.*

Being interaction-safe with respect to P_{R_1} and P_{R_2} will thus prevent the sharing of ground atoms of P_{R_1} and P_{R_2} by the ground instances of two conditionals R_1 and R_2. This will be used in the subsequent definition of the types of inter-rule interactions; this definition also does not refer to any constants in $D \setminus const(\mathcal{R})$.

Definition 7 (inter-rule interactions). *Let $R_1, R_2 \in \mathcal{R}$ be two different conditionals. Assume that $P_{R_1} \in at(R_1)$ and $P_{R_2} \in at(R_2)$ both use the same predicate symbol P, and that R_1 and R_2 are not interaction-safe with respect to P_{R_1} and P_{R_2}. Let U, V, W and Z be variables, $U \neq W$ and $V \neq Z$, and let c be a constant.*

1. *There is an* inter-rule interaction of type 1 *between R_1 and R_2 with respect to P, regarding the variable V and the constant c, denoted $R_1 \leftarrow \langle P\rangle_{V,c} \rightarrow R_2$, iff either*
   (a) $R_1 = \langle(\ldots P(\ldots, c, \ldots)\ldots)[\xi_{R_1}], C_{R_1}\rangle$, *and*
   $R_2 = \langle(\ldots P(\ldots, V, \ldots)\ldots)[\xi_{R_2}], C_{R_2}\rangle$ *and* $C_{R_2} \not\models V \neq c$[1], *or*
   (b) $R_1 = \langle(\ldots P(\ldots, U, \ldots)\ldots)[\xi_{R_1}], C_{R_1}\rangle$ *and* $C_{R_1} \models U \neq c$, *and*
   $R_2 = \langle(\ldots P(\ldots, V, \ldots)\ldots)[\xi_{R_2}], C_{R_2}\rangle$ *and* $C_{R_2} \not\models V \neq c$.
2. *There is an* inter-rule interaction of type 2 *between R_1 and R_2 with respect to P, regarding the variables V and Z, denoted $R_1 \leftarrow \langle P\rangle_{V,Z} \rightarrow R_2$, iff either*

[1] Note that if $C_{R_2} \models V \neq c$, then R_1 and R_2 would be interaction-safe with respect to P.

oarieli@mta.ac.il

(a) $R_1 = \langle(\ldots P(\ldots, U, \ldots, U, \ldots)\ldots)[\xi_{R_1}], C_{R_1}\rangle$, *and*
$R_2 = \langle(\ldots P(\ldots, V, \ldots, Z, \ldots)\ldots)[\xi_{R_2}], C_{R_2}\rangle$ *and* $C_{R_2} \nvDash V \neq Z$[2], *or*
(b) $R_1 = \langle(\ldots P(\ldots, U, \ldots, W, \ldots)\ldots)[\xi_{R_1}], C_{R_1}\rangle$ *and* $C_{R_1} \vDash U \neq W$, *and*
$R_2 = \langle(\ldots P(\ldots, V, \ldots, Z, \ldots)\ldots)[\xi_{R_2}], C_{R_2}\rangle$ *and* $C_{R_2} \nvDash V \neq Z$.

3. *Assume that* $Q_{R_1} \in atoms(R_1)$ *and* $Q_{R_2} \in atoms(R_2)$ *both use the same predicate symbol* Q*, and that* R_1 *and* R_2 *are not interaction-safe with respect to* Q_{R_1} *and* Q_{R_2}.[3] *There is an* inter-rule interaction of type 3 *between* R_1 *and* R_2 *with respect to* P *and* Q*, regarding the variables* V *and* Z*, denoted* $R_1 \leftarrow \langle P, Q\rangle_{V,Z} \rightarrow R_2$*, iff either*
(a) $R_1 = \langle(\ldots P(\ldots, U, \ldots)\ldots Q(\ldots, U, \ldots)\ldots)[\xi_{R_1}], C_{R_1}\rangle$, *and*
$R_2 = \langle(\ldots P(\ldots, V, \ldots)\ldots Q(\ldots, Z, \ldots)\ldots)[\xi_{R_2}], C_{R_2}\rangle$ *and* $C_{R_2} \nvDash V \neq Z$, *or*
(b) $R_1 = \langle(\ldots P(\ldots, U, \ldots)\ldots Q(\ldots, W, \ldots)\ldots)[\xi_{R_1}], C_{R_1}\rangle$ *and* $C_{R_1} \vDash U \neq W$, *and*
$R_2 = \langle(\ldots P(\ldots, V, \ldots)\ldots Q(\ldots, Z, \ldots)\ldots)[\xi_{R_2}], C_{R_2}\rangle$ *and* $C_{R_2} \nvDash V \neq Z$.

If there is an inter-rule interaction of type 1 or 2 between R_1 and R_2, there must be a predicate symbol P and corresponding atoms P_{R_1}, P_{R_2}, such that some, but not all the ground atoms of P_{R_1} resp. P_{R_2} are used by the ground instances of both R_1 and R_2, i.e., $GA_{P_{R_1}} \cap GA_{P_{R_2}} \neq \emptyset$ and $GA_{P_{R_1}} \neq GA_{P_{R_2}}$. Similarly, if there is an inter-rule interaction of type 3, there must be predicate symbols P, Q and corresponding atoms $P_{R_1}, Q_{R_1}, P_{R_2}, Q_{R_2}$, such that some, but not all of the combinations of ground atoms of P_{R_1}, Q_{R_1} resp. P_{R_2}, Q_{R_2} are used by the ground instances of both R_1 and R_2, i.e., $GA_{\langle P_{R_1}, Q_{R_1}\rangle} \cap GA_{\langle P_{R_2}, Q_{R_2}\rangle} \neq \emptyset$ and $GA_{\langle P_{R_1}, Q_{R_1}\rangle} \neq GA_{\langle P_{R_2}, Q_{R_2}\rangle}$.

For the other direction, a systematic case distinction over the structure of any R_1 and R_2 in $\mathcal{R}$ shows that there can not be any imbalanced sharing if there is no inter-rule interaction of type 1, 2 or 3 in $\mathcal{R}$, yielding:

Theorem 4 (Imbalanced sharing equivalent to inter-rule interaction). *There is an imbalanced sharing in* $\mathcal{R}$ *iff there is an inter-rule interaction of type 1, 2 or 3 in* $\mathcal{R}$.

6 Imbalanced Use and Intra-Rule Interactions

We will now investigate the reasons for an imbalanced use, and we will define intra-rule interactions as a correct and complete identification. Just like the inter-rule interactions defined above, the intra-rule interactions will also be independent of the constants occurring in $D \setminus const(\mathcal{R})$.

6.1 Reasons for an Imbalanced Use of Ground Atoms

From Example 4, we can identify two conditions for an imbalanced use of ground atoms of an atom Q_R by the ground instances of an FO-PCL conditional R: First, there

[2] Note that if $C_{R_2} \vDash V \neq Z$, then R_1 and R_2 would be interaction-safe with respect to P.
[3] Note that if Q was the same predicate symbol as P, then P_{R_1} and Q_{R_2} as well as Q_{R_1} and P_{R_2} would need to be considered for possible inter-rule interactions of type 1 or 2.

oarieli@mta.ac.il

is an inequality $U \neq V$ between a variable V occurring in Q_R, and another variable U occurring in an atom P_R but not in Q_R. Second, there is an additional restriction for the variable U. In Example 4, this additional restriction is an inequation, such that U may not be substituted by a specified constant a. This additional restriction causes an imbalance between those ground atoms of Q_R in which V is substituted by any constant $c \neq a$, and those ground atoms of Q_R in which V is substituted by a. This is because, if V is substituted by a, only one constant (a) is excluded for the substitution of U. But if V is substituted by any other constant $c \neq a$, two constants (a and c) are excluded for the substitution of U.

For the imbalance of ground atoms to occur, it is essential that the variable V (i.e., the variable that the additional constraint does *not* refer to) occurs in an atom that does not contain the variable U (i.e., the variable that the additional constraint refers to). This is because, if all the atoms that contain V also contain U, then the fact that there are more possible substitutions for U if $V = a$ than if $V \neq a$ can not cause an imbalance of ground atoms as substituting a different constant for U will always yield a different ground atom for all the atoms in which V occurs.

Example 8 (No imbalanced use of ground atoms[4]*).* Let $R = \langle (P(U)|Q(U,V))[\xi], U \neq V \wedge U \neq a \rangle$, and $P_R = P(U), Q_R = Q(U,V) \in at(R)$. The ground instances of R for the set of constants $D = \{a, b, c\}$ are:

$$gr_{R-1} = \langle (P(b)|Q(b,a)[\xi], \top \rangle \qquad gr_{R-3} = \langle (P(c)|Q(c,a)[\xi], \top \rangle$$
$$gr_{R-2} = \langle (P(b)|Q(b,c)[\xi], \top \rangle \qquad gr_{R-4} = \langle (P(c)|Q(c,b)[\xi], \top \rangle$$

The variable V does not occur in an atom that does not contain U. Substituting two different values for U will yield different ground atoms of Q_R, so there can not be an imbalance of ground atoms of Q_R.

In the following example, no specified constant a is excluded for the substitution of U (in fact, the conditional does not use any constant), but still there is an imbalance in the use of ground atoms of Q_R. Note that this situation with $const(\mathcal{R}) = \emptyset$ is not covered by the definition of an interaction in [10].

Example 9 (Imbalanced use of ground atoms). Let $R = \langle (P(U)|Q(V,Z))[\xi], U \neq V \wedge U \neq Z \rangle$, and $P_R = P(U), Q_R = Q(V,Z) \in at(R)$. The ground instances of R for the set of constants $D = \{a, b, c\}$ are:

$$gr_{R-1} = \langle (P(a)|Q(b,b))[\xi], \top \rangle \qquad gr_{R-7} = \langle (P(b)|Q(c,a))[\xi], \top \rangle$$
$$gr_{R-2} = \langle (P(a)|Q(b,c))[\xi], \top \rangle \qquad gr_{R-8} = \langle (P(b)|Q(c,c))[\xi], \top \rangle$$
$$gr_{R-3} = \langle (P(a)|Q(c,b))[\xi], \top \rangle \qquad gr_{R-9} = \langle (P(c)|Q(a,a))[\xi], \top \rangle$$
$$gr_{R-4} = \langle (P(a)|Q(c,c))[\xi], \top \rangle \qquad gr_{R-10} = \langle (P(c)|Q(a,b))[\xi], \top \rangle$$
$$gr_{R-5} = \langle (P(b)|Q(a,a))[\xi], \top \rangle \qquad gr_{R-11} = \langle (P(c)|Q(b,a))[\xi], \top \rangle$$
$$gr_{R-6} = \langle (P(b)|Q(a,c))[\xi], \top \rangle \qquad gr_{R-12} = \langle (P(c)|Q(b,b))[\xi], \top \rangle$$

There is an imbalance in the use of ground atoms of Q_R: The ground atoms $Q(a,a)$, $Q(b,b)$ and $Q(c,c)$ occur twice, while the ground atoms $Q(b,c)$, $Q(c,b)$, $Q(a,c)$, $Q(c,a)$, $Q(a,b)$, and $Q(b,a)$ occur only once. Because of this imbalance, no involution covering can be found.

[4] Compare this example to Example 4, in which an intra-rule interaction does occur.

oarieli@mta.ac.il

The two basic conditions we have identified for an imbalanced use of ground atoms are both present in Example 9: There is an inequality between a variable U and a variable V, and there is an additional constraint for U, which does not apply to V, in this case, an inequation between V and another variable Z. Therefore, if V is substituted with the same constant that Z is substituted with, there will be more possible substitutions for U than if V is substituted with a different constant than Z. In Example 9, this causes the effect that the ground instances of R contain more ground atoms of Q_R in which V and Z are substituted by the same constant than ground atoms of Q_R in which V and Z are substituted by different constants.

Example 10 (Removing the imbalance in Example 9). Continuing Example 9, we replace the conditional R with a conditional R_1, in which V is replaced by Z, which means that V and Z have to be substituted by the same constant, and a conditional R_2, in which V and Z may not be substituted by the same constant:

$$R_1 = \langle (P(U)|Q(Z,Z))[\xi], U \neq Z \rangle,$$
$$R_2 = \langle (P(U)|Q(V,Z))[\xi], U \neq V \wedge U \neq Z \wedge V \neq Z \rangle$$

Now, the set of ground instances $\{gr_{R-2}, gr_{R-3}, gr_{R-6}, gr_{R-7}, gr_{R-10}, gr_{R-11}\}$ belongs to R_1, and the set of ground instances $\{gr_{R-1}, gr_{R-4}, gr_{R-5}, gr_{R-8}, gr_{R-9}, gr_{R-12}\}$ belongs to R_2, and an involution covering can be found.

6.2 Reasons for an Imbalanced Use of Combinations of Ground Atoms

In the following example, although each single ground atom occurs the same number of times in the ground instances of R, there is an imbalance in the frequency of combinations of ground atoms.

Example 11 (Imbalanced use of combinations of ground atoms). Let $R = \langle (P(U)|Q(V) \wedge S(Z))[\xi], U \neq V \wedge U \neq Z \rangle$, and $P_R = P(U), Q_R = Q(V), S_R = S(Z) \in at(R)$. The ground instances of R for the set of constants $D = \{a, b, c\}$ are:

$$\begin{array}{ll}
gr_{R-1} = \langle (P(a)|Q(b) \wedge S(b))[\xi], \top \rangle & gr_{R-7} = \langle (P(b)|Q(c) \wedge S(a))[\xi], \top \rangle \\
gr_{R-2} = \langle (P(a)|Q(b) \wedge S(c))[\xi], \top \rangle & gr_{R-8} = \langle (P(b)|Q(c) \wedge S(c))[\xi], \top \rangle \\
gr_{R-3} = \langle (P(a)|Q(c) \wedge S(b))[\xi], \top \rangle & gr_{R-9} = \langle (P(c)|Q(a) \wedge S(a))[\xi], \top \rangle \\
gr_{R-4} = \langle (P(a)|Q(c) \wedge S(c))[\xi], \top \rangle & gr_{R-10} = \langle (P(c)|Q(a) \wedge S(b))[\xi], \top \rangle \\
gr_{R-5} = \langle (P(b)|Q(a) \wedge S(a))[\xi], \top \rangle & gr_{R-11} = \langle (P(c)|Q(b) \wedge S(a))[\xi], \top \rangle \\
gr_{R-6} = \langle (P(b)|Q(a) \wedge S(c))[\xi], \top \rangle & gr_{R-12} = \langle (P(c)|Q(b) \wedge S(b))[\xi], \top \rangle
\end{array}$$

Unlike in Example 9, the variables V and Z now appear in different atoms. Because of that, the overall number of ground atoms of each atom is balanced. Obviously, each ground instance of P_R, Q_R, and S_R appears exactly 4 times in the whole set of ground instances of R. However, there is an imbalance in the use of combinations of ground atoms of Q_R and S_R. For instance, the ground atom $Q(c)$ appears twice combined with the ground atom $S(c)$, but only once combined with $S(a)$ and $S(b)$, respectively. Because of that, no involution covering can be constructed for the set of ground instances.

In the case where the additional restriction for U is an inequation with a variable ($C \models U \neq V \wedge U \neq a$), we have found that an essential condition for an imbalance to occur

oarieli@mta.ac.il

is that there is an atom containing V (the variable that the additional constraint does *not* refer to) but not U (the variable that the additional constraint refers to). This condition applies here as well, as the following example shows.

Example 12 (No imbalanced use 1). Let $R = \langle (P(U)|Q(U,V) \wedge S(Z))[\xi], U \neq V \wedge U \neq Z \rangle$, and $P_R = P(U), Q_R = Q(U,V), S_R = S(Z) \in at(R)$. The ground instances of R for the set of constants $D = \{a,b,c\}$ are:

$$
\begin{array}{ll}
gr_{R-1} = \langle (P(a)|Q(a,b) \wedge S(b))[\xi], \top \rangle & gr_{R-7} = \langle (P(b)|Q(b,c) \wedge S(a))[\xi], \top \rangle \\
gr_{R-2} = \langle (P(a)|Q(a,b) \wedge S(c))[\xi], \top \rangle & gr_{R-8} = \langle (P(b)|Q(b,c) \wedge S(c))[\xi], \top \rangle \\
gr_{R-3} = \langle (P(a)|Q(a,c) \wedge S(b))[\xi], \top \rangle & gr_{R-9} = \langle (P(c)|Q(c,a) \wedge S(a))[\xi], \top \rangle \\
gr_{R-4} = \langle (P(a)|Q(a,c) \wedge S(c))[\xi], \top \rangle & gr_{R-10} = \langle (P(c)|Q(c,a) \wedge S(b))[\xi], \top \rangle \\
gr_{R-5} = \langle (P(b)|Q(b,a) \wedge S(a))[\xi], \top \rangle & gr_{R-11} = \langle (P(c)|Q(c,b) \wedge S(a))[\xi], \top \rangle \\
gr_{R-6} = \langle (P(b)|Q(b,a) \wedge S(c))[\xi], \top \rangle & gr_{R-12} = \langle (P(c)|Q(c,b) \wedge S(b))[\xi], \top \rangle
\end{array}
$$

The fact that there are more possible substitutions for U if V and Z are substituted by the same constant can not cause an imbalance in combinations of ground atoms of Q_R and S_R, because two different substitutions for U will always yield different ground atoms for Q_R. Therefore, it is not possible that a combination of ground atoms of Q_R and S_R appears more often than another one.

As we are dealing with 3 variables here, we also have to consider the position of the variable Z. It actually does not suffice that V appears in an atom that does not contain U, but the same is true for Z: If each atom S_R that contains Z also contains U, then two different substitutions for U will also yield a different ground atom for S_R. Therefore, it is impossible that a combination of S_R with the atom that uses V appears more often than another one.

Example 13 (No imbalanced use 2). Let $R = \langle (P(U)|Q(V) \wedge S(U,Z))[\xi], U \neq V \wedge U \neq Z \rangle$, and $P_R = P(U), Q_R = Q(V), S_R = S(U,Z) \in at(R)$. The ground instances of R for the set of constants $D = \{a,b,c\}$ are:

$$
\begin{array}{ll}
gr_{R-1} = \langle (P(a)|Q(b) \wedge S(a,b))[\xi], \top \rangle & gr_{R-7} = \langle (P(b)|Q(c) \wedge S(b,a))[\xi], \top \rangle \\
gr_{R-2} = \langle (P(a)|Q(b) \wedge S(a,c))[\xi], \top \rangle & gr_{R-8} = \langle (P(b)|Q(c) \wedge S(b,c))[\xi], \top \rangle \\
gr_{R-3} = \langle (P(a)|Q(c) \wedge S(a,b))[\xi], \top \rangle & gr_{R-9} = \langle (P(c)|Q(a) \wedge S(c,a))[\xi], \top \rangle \\
gr_{R-4} = \langle (P(a)|Q(c) \wedge S(a,c))[\xi], \top \rangle & gr_{R-10} = \langle (P(c)|Q(a) \wedge S(c,b))[\xi], \top \rangle \\
gr_{R-5} = \langle (P(b)|Q(a) \wedge S(b,a))[\xi], \top \rangle & gr_{R-11} = \langle (P(c)|Q(b) \wedge S(c,a))[\xi], \top \rangle \\
gr_{R-6} = \langle (P(b)|Q(a) \wedge S(b,c))[\xi], \top \rangle & gr_{R-12} = \langle (P(c)|Q(b) \wedge S(c,b))[\xi], \top \rangle
\end{array}
$$

The fact that there are more possible substitutions for U if V and Z are substituted by the same constant can not cause an imbalance in combinations of ground atoms of Q_R and S_R, because two different substitutions for U will always yield different ground atoms for S_R. Therefore, it is not possible that a combination of ground atoms of Q_R and S_R appears more often than another one.

6.3 Intra-Rule Interactions

Having analyzed what causes an imbalanced use of ground atoms or combinations of ground atoms, we define intra-rule interactions as a correct and complete identification.

oarieli@mta.ac.il

Definition 8 (intra-rule interactions). *Let $R = \langle(\phi_R|\psi_R)[\xi_R], C_R\rangle \in \mathcal{R}$ be a conditional in constraint normal form.*[5] *Let $P_R, Q_R \in at(R)$, using the predicate symbol P respectively Q, with $Q_R \neq P_R$.*[6] *Let U, V and Z be different variables and c a constant.*

1. *There is an* intra-rule interaction of type 1 *in R with respect to Q, regarding the variable V and the constant c, denoted $\langle Q\rangle_{V,c} \rightarrow R$, iff*
   $P_R = P(\ldots, U, \ldots)$, $Q_R = Q(\ldots, V, \ldots)$,
   $U \notin vars(Q_R)$, $C_R \vDash U \neq V$, $C_R \vDash U \neq c$, *and* $C_R \nvDash V \neq c$.
2. *There is an* intra-rule interaction of type 2 *in R with respect to Q, regarding the variables V and Z, denoted $\langle Q\rangle_{V,Z} \rightarrow R$, iff*
   $P_R = P(\ldots, U, \ldots)$, $Q_R = Q(\ldots, V, \ldots, Z \ldots)$,
   $U \notin vars(Q_R)$, $C_R \vDash U \neq V$, $C_R \vDash U \neq Z$, *and* $C_R \nvDash V \neq Z$.
3. *Let $S_R \in at(R)$ use the predicate symbol S, $S_R \neq P_R$ and $S_R \neq Q_R$. There is an* intra-rule interaction of type 3 *in R with respect to Q and S, regarding the variables V and Z, denoted $\langle Q, S\rangle_{V,Z} \rightarrow R$, iff*
   $P_R = P(\ldots, U, \ldots)$, $Q_R = Q(\ldots, V, \ldots)$, $S_R = S(\ldots Z \ldots)$,
   $U \notin vars(Q_R)$, $U \notin vars(S_R)$, $C_R \vDash U \neq V$, $C_R \vDash U \neq Z$, *and* $C_R \nvDash V \neq Z$.

Every intra-rule interaction of type 1 or 2 in a conditional R causes an imbalanced use of ground atoms in the ground instances of R, and every intra-rule interaction of type 3 causes an imbalanced use of combinations of ground atoms in the ground instances of R. Conversely, a systematic case distinction over the possible structure of a conditional R reveals that if there is no intra-rule interaction of type 1, 2, or 3 in R, then there is no imbalanced use in R, giving us the following theorem.

Theorem 5 (Imbalanced use equivalent to intra-rule interaction). *There is an imbalanced use in $\mathcal{R}$ iff there is an intra-rule interaction of type 1, 2 or 3 in $\mathcal{R}$.*

7 A Complete Set of Transformation Rules

In Examples 5 and 10 we already illustrated how a conditional preventing parametric uniformity due to an imbalance could be replaced. Exploiting the notions of inter-rule and intra-rule interactions elaborated above, Figure 1 presents a set $\mathcal{PU}$ of rules that transforms any knowledge base $\mathcal{R}$ into an equivalent knowledge base $\mathcal{R}'$ that is parametrically uniform. The notation $\overline{\sigma}(R)$ for a substitution $\sigma = \{X/Y\}$ used in $\mathcal{PU}$ denotes the conditional obtained from R by adding $X \neq Y$ to its constraint formula.

Any rule in $\mathcal{PU}$ splits a conditional R into $\sigma(R)$ and $\overline{\sigma}(R)$ with disjoint sets of ground instances, their union being the set of ground instances of R, and therefore:

Theorem 6 (Correctness of $\mathcal{PU}$). *Let $\mathcal{R}$ be an FO-PCL knowledge base. Then applying any transformation rule from $\mathcal{PU}$ to $\mathcal{R}$ yields an FO-PCL knowledge base $\mathcal{R}'$ having the same models. Thus, in particular, $\mathcal{R}$ and $\mathcal{R}'$ have the same maximum entropy model.*

[5] Note that the following conditions will not apply if $C_R = \bot$, thus R is constraint-consistent.
[6] Note that while P_R and Q_R are different atoms, P and Q might be the same predicate symbol.

oarieli@mta.ac.il

$$(TE_1) \quad \frac{\mathcal{R} \cup \{R_1, R_2\}}{\mathcal{R} \cup \{R_1\} \cup \nu\{\sigma(R_2), \overline{\sigma}(R_2)\}} \qquad R_1 \leftarrow \langle P \rangle_{V,c} \rightarrow R_2, \;\; \sigma = \{V/c\}$$

$$(TE_2) \quad \frac{\mathcal{R} \cup \{R_1, R_2\}}{\mathcal{R} \cup \{R_1\} \cup \nu\{\sigma(R_2), \overline{\sigma}(R_2)\}} \qquad R_1 \leftarrow \langle P \rangle_{V,Z} \rightarrow R_2, \;\; \sigma = \{V/Z\}$$

$$(TE_3) \quad \frac{\mathcal{R} \cup \{R_1, R_2\}}{\mathcal{R} \cup \{R_1\} \cup \nu\{\sigma(R_2), \overline{\sigma}(R_2)\}} \qquad R_1 \leftarrow \langle P, Q \rangle_{V,Z} \rightarrow R_2, \;\; \sigma = \{V/Z\}$$

$$(TA_1) \quad \frac{\mathcal{R} \cup \{R\}}{\mathcal{R} \cup \nu\{\sigma(R), \overline{\sigma}(R)\}} \qquad \langle Q \rangle_{V,c} \rightarrow R, \;\; \sigma = \{V/c\}$$

$$(TA_2) \quad \frac{\mathcal{R} \cup \{R\}}{\mathcal{R} \cup \nu\{\sigma(R), \overline{\sigma}(R)\}} \qquad \langle Q \rangle_{V,Z} \rightarrow R, \;\; \sigma = \{V/Z\}$$

$$(TA_3) \quad \frac{\mathcal{R} \cup \{R\}}{\mathcal{R} \cup \nu\{\sigma(R), \overline{\sigma}(R)\}} \qquad \langle Q, S \rangle_{V,Z} \rightarrow R, \;\; \sigma = \{V/Z\}$$

Fig. 1. Transformation rules $\mathcal{PU}$ yielding parametric uniformity

Since the conditions for inter- and intra-rule interactions are mirrored in the conditions of the transformation rules in $\mathcal{PU}$, we have:

Theorem 7 **($\mathcal{PU}$ removes all interactions).** *Let $\mathcal{R}$ be a FO-PCL knowledge base. If no transformation rule from $\mathcal{PU}$ is appcicable to $\mathcal{R}$, then $\mathcal{R}$ does not contain any inter-rule or intra-rule interaction.*

The preconditions of any of the transformation rules in $\mathcal{PU}$ ensure that the splitting of a constraint-consistent conditional R' yields conditionals $\sigma(R')$ and $\overline{\sigma}(R')$ that are again constraint-consistent:

Theorem 8 **($\mathcal{PU}$ preserves constraint-consistency).** *Let $\mathcal{R}$ be a constraint-consistent FO-PCL knowledge base. Then applying any transformation rule from $\mathcal{PU}$ to $\mathcal{R}$ yields an FO-PCL knowledge base that is constraint-consistent.*

In each application of a transformation rule from $\mathcal{PU}$, the number of different variables occurring in the premise or conclusion of the splitted conditional is decreased or the number of constraints on the variables is increased. Since no new variables are introduced, $\mathcal{PU}$ can not be applied indefinitely.

Theorem 9 **(Termination of $\mathcal{PU}$).** *Exhaustively applying $\mathcal{PU}$ to an FO-PCL knowledge base $\mathcal{R}$ always terminates.*

Putting the obtained results together yields that $\mathcal{PU}$ is correct and complete:

Theorem 10 **($\mathcal{PU}(\mathcal{R})$ is parametrically uniform).** *Exhaustively applying $\mathcal{PU}$ to a knowledge base $\mathcal{R}$ yields a knowledge base $\mathcal{PU}(\mathcal{R})$ such that $\mathcal{R}$ and $\mathcal{PU}(\mathcal{R})$ have the same maximum-entropy model and $\mathcal{PU}(\mathcal{R})$ is parametrically uniform.*

As pointed out before, inter- and intra-rule interactions and thus also the rules in $\mathcal{PU}$ are independent of the set of constants occurring in $D \setminus const(\mathcal{R})$. It can be shown that $\mathcal{PU}(\mathcal{R})$ is parametrically uniform w.r.t. any set of constants D' with $const(\mathcal{R}) \subseteq D'$.

oarieli@mta.ac.il

8 Related Work and Conclusions

In relational probabilistic logic, instantiating conditionals may readily lead to inconsistent instances. In the relational probabilistic conditional logic FO-PCL, such inconsistences can be avoided by using constraints restricting the possible instantiations of a conditional to admissable ones. In this paper, we developed a method for transforming any FO-PCL knowledge base $\mathcal{R}$ into an equivalent one that is parametrically uniform. Parametric uniformity provides an approach to lifted inference in FO-PCL since it ensures that for all instances of a conditional only one single maximum entropy optimization parameter has to be determined.

The splitting operation used in [19] is related to the replacement of conditionals used in our transformation rules, but [19] employs Bayesian networks and draws no connection to the maximum entropy principle FO-PCL semantics is based upon. Maximum entropy semantics have also been used in probabilistic logic programming (see e.g. [13,6,23,3]) where often an interval of probabilities is employed instead of a single probability in order to take different instantiations into account.

For another variant of maximum entropy semantics for relational probabilistic logic, the possible computational benefits of lifted inference when all predicates are unary are studied in [21]. In [22], FO-PCL and two other relational maximum entropy semantics, *aggregating* and *averaging* semantics, are compared to each other. However, the constraint formulas of FO-PCL avoiding inconsistent groundings are not present in these other approaches; instead, certain derivations from the probabilities given in the conditionals are taken into account by considering groundings determined by the underlying universe [22]. Some general evaluation and comparison criteria for approaches to probabilistic relational knowledge representation can be found in [2].

While we have shown that the transformation rules $\mathcal{PU}$ yield a parametrically uniform knowledge base $\mathcal{R}'$ for any $\mathcal{R}$, in this paper we did not address the question of how the number of conditionals in $\mathcal{R}$ and $\mathcal{R}'$ are related. Whereas $\mathcal{R}'$ may contain instantiations of conditionals involving constants from $const(\mathcal{R})$, it is still fully generic with respect to $D \setminus const(\mathcal{R})$ since $\mathcal{PU}$ is independent of these constants. We are currently developing a software system implementing the transformation rules, providing a platform for evaluating $\mathcal{PU}$ with respect to various knowledge bases. Our current work also includes the investigation of confluence properties of our approach and how it might carry over to other relational maximum entropy semantics.

References

1. Adams, E.W.: The Logic of Conditionals. D. Reidel Publishing Company, Dordrecht (1975)
2. Beierle, C., Finthammer, M., Kern-Isberner, G., Thimm, M.: Evaluation and Comparison Criteria for Approaches to Probabilistic Relational Knowledge Representation. In: Bach, J., Edelkamp, S. (eds.) KI 2011. LNCS (LNAI), vol. 7006, pp. 63–74. Springer, Heidelberg (2011)
3. Broecheler, M., Simari, G.I., Subrahmanian, V.S.: Using Histograms to Better Answer Queries to Probabilistic Logic Programs. In: Hill, P.M., Warren, D.S. (eds.) ICLP 2009. LNCS, vol. 5649, pp. 40–54. Springer, Heidelberg (2009)
4. Delgrande, J.P.: On First-Order Conditional Logics. Artificial Intelligence 105(1-2), 105–137 (1998)

oarieli@mta.ac.il

5. Fagin, R., Halpern, J.Y.: Reasoning about knowledge and probability. J. ACM 41(2), 340–367 (1994)
6. Fagin, R., Halpern, J.Y., Megiddo, N.: A logic for reasoning about probabilities. Information and Computation 87(1-2), 78–128 (1990)
7. Fisseler, J.: Learning and Modeling with Probabilistic Conditional Logic. Dissertations in Artificial Intelligence, vol. 328. IOS Press, Amsterdam (2010)
8. Getoor, L., Taskar, B. (eds.): Introduction to Statistical Relational Learning. MIT Press (2007)
9. Halpern, J.Y.: Reasoning About Uncertainty. MIT Press (2005)
10. Janning, R., Beierle, C.: Transformation Rules for First-Order Probabilistic Conditional Logic Yielding Parametric Uniformity. In: Bach, J., Edelkamp, S. (eds.) KI 2011. LNCS (LNAI), vol. 7006, pp. 157–168. Springer, Heidelberg (2011)
11. Kern-Isberner, G.: Characterizing the principle of minimum cross-entropy within a conditional-logical framework. Artificial Intelligence 98, 169–208 (1998)
12. Kern-Isberner, G.: Conditionals in Nonmonotonic Reasoning and Belief Revision. LNCS, vol. 2087, p. 190. Springer, Heidelberg (2001)
13. Kern-Isberner, G., Lukasiewicz, T.: Combining probabilistic logic programming with the power of maximum entropy. Artificial Intelligence, Special Issue on Nonmonotonic Reasoning 157(1-2), 139–202 (2004)
14. Krämer, A.: Transformation rules for lifted inference in relational probabilistic logic knowledge bases. B.Sc. Thesis, Dept. of Computer Science, FernUniversität in Hagen, Germany (2011)
15. Nilsson, N.J.: Probabilistic logic. Artif. Intell., 71–87 (1986)
16. Nute, D., Cross, C.B.: Conditional logic. In: Gabbay, D.M., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. 4, pp. 1–98. Kluwer Academic Publishers (2002)
17. Paris, J.B.: The uncertain reasoner's companion - A mathematical perspective. Cambridge University Press (1994)
18. Paris, J.B., Vencovska, A.: In defence of the maximum entropy inference process. International Journal of Approximate Reasoning 17(1), 77–103 (1997)
19. Poole, D.: First-order probabilistic inference. In: Gottlob, G., Walsh, T. (eds.) Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), pp. 985–991. Morgan Kaufmann (2003)
20. Rödder, W., Kern-Isberner, G.: Representation and extraction of information by probabilistic logic. Information Systems 21(8), 637–652 (1996)
21. Thimm, M.: Probabilistic Reasoning with Incomplete and Inconsistent Beliefs. Dissertation, Technische Universität Dortmund (2011)
22. Thimm, M., Kern-Isberner, G., Fisseler, J.: Relational Probabilistic Conditional Reasoning at Maximum Entropy. In: Liu, W. (ed.) ECSQARU 2011. LNCS, vol. 6717, pp. 447–458. Springer, Heidelberg (2011)
23. Yue, A., Liu, W., Hunter, A.: Measuring the Ignorance and Degree of Satisfaction for Answering Queries in Imprecise Probabilistic Logic Programs. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 386–400. Springer, Heidelberg (2008)

oarieli@mta.ac.il

Flexible and Efficient Distributed Resolution of Large Entities*

András J. Molnár, András A. Benczúr, and Csaba István Sidló

Data Mining and Web Search Group, Informatics Laboratory
Institute for Computer Science and Control, Hungarian Academy of Sciences
{modras, benczur, sidlo}@ilab.sztaki.hu

Abstract. Entity resolution (ER) is a computationally hard problem of data integration scenarios, where database records have to be grouped according to the real-world entities they belong to. In practice these entities may consist of only a few records from different data sources with typos or historical data. In other cases they may contain significantly more records, especially when we search for entities on a higher level of a concept hierarchy than records.

In this paper we give theoretical foundation of a variety of practically important match functions. We show that under these formulations, ER with large entities can be solved efficiently with algorithms based on MapReduce, a distributed computing paradigm. Our algorithm can efficiently incorporate probabilistic and similarity-based record match, enabling flexible match function definition. We demonstrate the usability of our model and algorithm in a real-world insurance ER scenario, where we identify household groups of client records.

1 Introduction

Entity Resolution (ER) is the process of identifying groups of records that refer to the same real-world entity. The process was described in several contexts under many different names: duplicate detection, instance identification, heterogeneous join, merge/purge, reference reconciliation, or object matching. Closely related topics include clustering, similarity join, string similarity, data cleaning, data warehousing, data integration and information integration.

In most cases, records are heterogeneous and erroneous and hence the mapping to hidden real-world entities is not straightforward. Structural and syntactic heterogeneity originates mostly from the heterogeneity of source systems, difference in data handling policies, standards, and finally from low data quality due to typos, missing values and other problems. ER can be therefore handled as a data cleansing task, occurring in data integration scenarios often.

Entity resolution is an actively researched area, and the problem can be formulated in many different ways. Input and output can be a set of records with attributes, a set of XML documents or a graph. The algorithms can either produce exact results or probabilistic mappings. Match functions can be defined by

* This work was supported by the EU FP7 SEC project SCIIMS (Ref. 218223).

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 244–263, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

exact rules, by similarities or by links between records. Results can be represented by record sets, by representative merged elements, or both. Training data or entity activity log can be present. The architecture used to solve the problem can be distributed, can be a single database server, a standard standalone computer or, for another example, a data mining framework.

Our main contributions are the following. We present an extended data model and problem formulation based on indexable features that facilitate the formulation of the given business logic, concentrating on match functions of independent entity properties, and providing a framework for defining efficient indexes. Based on the model we describe a scalable distributed algorithm for MapReduce. The algorithm is able to scale up to hundreds of millions of records and copes with large entities as well.

The rest of this paper is organized as follows. After giving an overview of the related work, we describe a motivating insurance ER scenario and enumerate issues of creating a client database. We formally define the ER problem in Section 3 based on the concept of indexable features. Our distributed algorithm is described in Section 5. Techniques to define indexable features for efficient resolution are described in Section 6. Finally, evaluation of the proposed methods is given in Section 7.

1.1 Related Work

One of the first descriptions of record linkage appears in the influential paper of Fellegi and Sunter [18] in 1969, describing a probabilistic model. Since then, entity resolution problems have been studied in many different disciplines and names. For overview, in [17] a survey is given on duplicate record detection, describing supervised, unsupervised and active learning, and summarizing statistical and machine learning solutions based on various text similarity and match measures. Recently the book [38] introduces key models, methods and new trends from a more practical point of view.

Traditional deduplication approach uses similarity measures for attributes, and learns when two records can be resolved to the same entity. A survey of string similarity functions can be found in [17], along with a survey of basic duplicate detection algorithms. In [20] a nice solution is presented for implementing string-similarity joins using q-grams in an RDBMS environment.

ER can be handled as a supervised learning problem, if training data is present. We can apply data mining classification methods, for example Bayes methods [23,18], decision trees [31] or SVM [7,11]. Unsupervised learning methods such as latent Dirichlet allocation [3] or clustering methods can also be used, if there is no training data. An interesting approach lying between the previous two is called active learning: when a small set of training data is given, the algorithm decides the new elements it could use the best to extend the training set ([34]). An automated training data selection method is described in [28]. Recently, [10] shows that cost sensitive alternating decision trees are practical for industrial applications weighting type I and II errors, while producing easily interpretable models.

oarieli@mta.ac.il

ER is formalized many times as generating clusters of linked records. In the citation database scenario, with the goal of identifying authors, we do not really have author attributes other than their names. We can however link these records by joint papers. This way ER can be seen as a special problem of link-mining; a survey containing link based entity resolution can be found in [19]. The approach is called relational ER, based on the relations between records, or collective ER, because we would like to resolve records based on the link graph as a whole.

Entity resolution as a hypergraph clustering problem can be found in [4], under the name of relational clustering. Input data is handled as a reference graph, with nodes as entity records and edges as links between these nodes. The goal of the resolution process is to produce a resolved entity graph, where nodes are entity instances that hold entity records together. Clustering is also suggested [24]; however, general clustering methods are usually designed for larger and less number of clusters than records of entities in ER.

A seminal paper, [2] (published first in 2005) introduces generic entity resolution with black-box match and merge functions, where resolution means the closure of the original entity set according to these functions. Simple feature indexes are also used. The model and the algorithms are extended in [32] for handling approximate results as records with confidences. [1] adapts the algorithms to a distributed environment. Our generic ER algorithms for relational databases were published in [35].

Other interesting approaches to ER include utilizing aggregate constraints [9], or giving methods for query time ER [5]. In [6] a unified model is suggested for entity identification and document categorization. [43] widens the coreference problem with schema matching and canonization, and provides a unified model. The role of cross-field dependencies is described in detail in [22].

Recently, several new ER results were published. A new approach can be found in [44]: entity behavior is recorded as a transactional log. Common patterns of these transactions are used to identify similar or identical entities. Measuring the quality of entity resolution results is a crucial problem [33], dealing with possible quality metrics. [41] enhances core ER algorithms by combining the results of different blocking strategies. [21] exploits the role of constraints when finding duplicates. [40] deals with the effect of match/merge rule evolution, and gives methods to preserve results when rules change. [14] builds special inverted indexes to speed up ER with blocking. A survey of indexing techniques available for deduplication is provided in [12], including blocking, sorted neighborhood, Q-grams and canopies.

Entity resolution frameworks including SERF, MTB, DDUpe and MARLIN (see [29] for a survey) integrate several variations of the problem formulation in effective systems. A practical comparison of ER approaches can be found in [30] using the FEVER framework. The Febrl framework also provides parallelization [13]. Other parallel algorithms are presented in [26], tested on a few thousands of records. More recently [27] introduces parallel match and a distributed infrastructure, using similarity-based matchers.

oarieli@mta.ac.il

In [36] we studied how efficient indexing methods can be used to speed up the ER process. Our first results on scalable parallel ER were published in [37].

2 Motivating Example

Companies typically face the entity resolution problem when building a client database, or manage client master data. Clients may appear multiple times in multiple source systems, e.g. a record for a contract, another for a purchase. As another example, the same person may appear in several marketing databases obtained by different means. ER is the key step in producing sound and clean client master data.

Our motivating application is the client data integration of several insurance source systems at AEGON Hungary Insurance Ltd.[1] The ER problem comes into sight during the construction of a client data mart over legacy systems that remained independent of each other for operational reasons during merges and ownership changes.

Client records may consist of attributes, both of persons (birth data, tax and social security numbers, postal address, etc.) and of organizations (client ID, contract number). Attribute values are often missing or erroneous, and some attributes change in time (name, postal address).

In addition to finding records of clients, another useful ER task is to find household entities. The use of households enable efficient marketing and a better knowledge of clients. The task requires however more complex match strategies, using for example postal addresses or phone numbers to deduce relationships between clients. Furthermore, household entities contain more records in average than usual ER subjects: usually more than one client belongs to a household.

Match between records might be based on pairwise equality of one or more attributes. There are cases, however, when the definition of match is more complex. Some of the attributes might refer to the same concept with possible cross-matches (e.g. home address and postal address, general phone number and mobile phone, or a name and a maiden name). There might be frequent, dummy phone numbers or incomplete addresses that produce false merges unless we can use some filtering, for instance, based on frequencies in a probabilistic model. Some records might be ambiguous and belong to more than one entity at the same time. Some of the overlapping entities might be merged, some others might be not, based on the desired match logic of the actual entity resolution task.

Figure 1 depicts such an overlapping situation. Nodes represent data records, containing customer names and addresses with edges showing the match relationship. Sets correspond to discovered households as entities. The two examples demonstrate that the match logic in such a case must be more explicitly defined and is not implied by the pairwise matches alone: the entities on the left can be merged while on the right not.

[1] AEGON Hungary has been a member of the AEGON Group since 1992, one of the world's largest life insurance and pension groups, and a strong provider of investment products.

oarieli@mta.ac.il

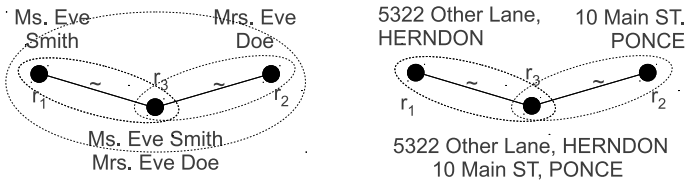


Fig. 1. An example of overlapping entities

In the first case the married name is different from the maiden name. Record r_1 contains maiden name only, while r_2 the current name and r_3 both names. Other attributes might also be equal, giving enough confidence for match except between r_1 and r_2 due to the lack of a common name. In this case, we can conclude that these three records belong to the same person and hence, the same household, so the records can be unified as a single entity.

In the second case we see the addresses of the same person with the address of the parents in r_1 and of the husband in r_2. We have two overlapping households: one with the parents and another, distinct one with the husband. In such a case, the three records can not be unified.

This simple match pattern demonstrates the motivation behind considering the entities as record sets instead of single representative records and the match relation to be defined over the sets instead of record pairs only. More complex match logic can be treated in this way, as well. In the following, we give a more general model, which can serve as a basis for different entity resolution approaches, one of them being the reachable subset model we introduce later.

3 Problem Formulation

3.1 The General Model

Let a set of **records** be $R = \{r_1, r_2, ...r_m\}$, where each r_j is described by its k **attribute values** $a_{r_j 1}, \ldots, a_{r_j k}$ such as ID, name, address etc. Each record has the same k number attributes. Some attribute values may be missing, e.g. we may not know the e-mail address of customer j, that we denote by $a_{r_j \ell} = \emptyset$ (NULL value).

More formally, for a fixed k and for each $i \in [1..k]$ the ith **attribute** is a function $a_{\cdot i} : R \to DOM_i^* = DOM_i \cup \{\emptyset\}$, where DOM_i can be any set as the attribute domain. $a_{\cdot i}(r)$ is denoted as a_{ri} for simplicity.

For a general formulation of our problem we introduce a partial algebra with a binary merge operation $\langle \cdot, \cdot \rangle$ over R and a binary match relation $\sim$:

$$\mathcal{E} = (E, \langle \cdot, \cdot \rangle, \sim)$$

The base set E of this partial algebra is called the set of **entities**. The algebra is generated by a subset $R \subseteq E$ called the record set R.

oarieli@mta.ac.il

The match relation $\sim$ is a general construct on entities, and might be defined arbitrarily. In practice it is defined according to the actual entity resolution task, and is based on or related to the attributes of records. We introduce features as a practical way to define entity matching in Section 4.

The base set E is generated by R, i.e. we may define

$$E_0 = R, \quad E_i = E_{i-1} \cup \{\langle x, y\rangle | x, y \in E_{i-1} \wedge x \sim y\}, \quad E = \bigcup_{i=0}^{\infty} E_i.$$

Let the **entity resolution** $ER(\mathcal{E})$ be the null elements of E, i.e. the set of all "maximal" elements of E that cannot be extended by merging matching elements:

$$ER(\mathcal{E}) = \{x \in E \mid \forall y \in E : (y \sim x \Rightarrow \langle y, x\rangle = x) \wedge (x \sim y \Rightarrow \langle x, y\rangle = x)\}$$

The base set E is not necessarily finite and maximal elements may also be generated by applying the $\langle , \rangle$ operator infinitely many times. This may happen if we concatenate values of the attributes without removing duplicates, growing attribute values longer and longer.

As another example, E may not contain null elements at all, i.e. there may be no maximal, non-extendible entities in the data. A (cyclic) group is clearly an algebraic example; the corresponding ER task keeps the most recently added attribute value a_1 of x_1 and a_2 of x_2 when defining $\langle , \rangle$. In this case

$$\langle \ldots \langle\langle\langle x_1, x_2\rangle, x_1\rangle, x_2\rangle \ldots x_i\rangle \tag{1}$$

is equal to the last element $x_i = x_1$ or x_2 of the expression.

Different algebras may result in different models of entity resolution. For instance, the model used by [2] corresponds to the setting $E \subseteq \times_{i=1}^{k} DOM_i^*$, so that the merge operation maps to single, unified representative records. It also introduces the Swoosh algorithm and the so-called ICAR conditions for the merge operation and match relation to make sure the algorithm works correctly.

The definition for ER above, as shown by the examples, is too general for practice. Next we introduce further constraints to reduce the complexity of the general ER problem.

3.2 The ICAR Conditions and Their Limitation

The ICAR conditions of [2] are considered a standard definition of the entity resolution problem. After giving the definition we will show that the fourth representativity condition is too strong and introduces inflexibility for practical applications, especially if entity boundaries are not sharp and may overlap.

Idempotence: For all r, $r \sim r$ and $\langle r, r\rangle = r$. A record always matches itself, and merging it with itself still yields the same record.

Commutativity: For all r_1, r_2, $r_1 \sim r_2$ iff $r_2 \sim r_1$, and if $r_1 \sim r_2$, then $\langle r_1, r_2\rangle = \langle r2, r1\rangle$.

oarieli@mta.ac.il

(Weak) Associativity: For all $r_1, r_3 \in R$ and $X \in E$ such that $\langle\langle r_1, X\rangle, r_3\rangle$ and $\langle r_1, \langle X, r_3\rangle\rangle$ exist, $\langle\langle r_1, X\rangle, r_3\rangle = \langle r_1, \langle X, r_3\rangle\rangle$. This constraint is required for records X only in [2].

Representativity: If $r_3 = \langle r1, r2\rangle$, then for any r_4 such that $r_1 \sim r4$, we also have $r_3 \sim r_4$.

Representativity is too strong in that it does not allow overlapping entities. In the example of households, we may however have a household consisting of a person with parent and another distinct one with spouse, a practically important case of overlapping entities.

Although the original paper [2] defines the third property as associativity, it is in fact weaker than the usual definition that also requires the existence of $\langle r_1, \langle r_2, r_3\rangle\rangle$, whenever $\langle\langle r_1, r_2\rangle, r_3\rangle$ exist. The stronger version of associativity does not hold in practice either, since r_1 may share an attribute with r_3 but may be distinct from r_2. In the example of Fig. 1, r_1 may not be similar to r_2 while $\langle\langle r_1, r_2\rangle, r_3\rangle$ may exist and correspond to the person as an entity.

Also note that Idempotence and Commutativity implies the reflexivity and symmetry of the $\sim$ relation. Note that if associativity is further strengthened to imply transitivity, the match relation becomes an equivalence of the records.

Instead of associativity and as a weaker version of representativity, a better motivated definition that can be easily checked over the example of Fig. 1 is

$$\begin{aligned} \langle X, r_2\rangle \sim r_3 \text{ implies either } \quad & X \sim r_2 \text{ and } \langle\langle X, r_2\rangle, r_3\rangle = \langle X, \langle r_2, r_3\rangle\rangle, \text{ or} \\ & X \sim r_3 \text{ and } \langle\langle X, r_2\rangle, r_3\rangle = \langle r_2, \langle X, r_3\rangle\rangle. \end{aligned} \tag{2}$$

Under the additional constraints in the next section, the definition in (3) is equivalent to weak associativity. For this reason, we give no motivation for weak associativity, simply use as a notion from [2].

3.3 The Accessible Subset Model

For our algorithms, we introduce the accessible subset model as an alternative approach, which allows us to relax some of the ICAR conditions. Our model will have a similar but weaker implication to form representative merged elements: as in ICAR, the merge operation is simply the union of record sets. The concept below is hence is a weaker version of the representativity from the ICAR properties in [2].

For $X \in E$, let rank (X) be the minimum number of generator elements R, with multiplicities, needed to generate X. We define the *accessibility* property similar to [8] as follows:

$$\forall X \in E \exists r \in R, Y \in E \text{ with } \mathrm{rank}(Y) < \mathrm{rank}(Y) \text{ such that } X = \langle y, r\rangle. \tag{3}$$

We extend the absorption property

$$\langle\langle r_1, r_2\rangle, r_1\rangle = \langle r_1, r_2\rangle \tag{4}$$

oarieli@mta.ac.il

and define **strong absorptivity**: if two elements of E are generated by the same set of records R, then they are equal.

By the above requirements, the elements E of the partial algebra becomes a partial lattice of records and henceforth we may simply consider subsets of R instead of general expressions generated by R:

Theorem 1. *If $(E, \langle\cdot,\cdot\rangle, \sim)$ is idempotent, commutative, weak associative, accessible and strong absorptive, then E is in bijection with 2^R (subsets of R) and if $\langle\cdot,\cdot\rangle$ exists, then it is equal to the set union.*

Proof. We only have to show that every $X \in E$ can be generated as

$$\langle \ldots \langle\langle r_{i_1}, r_{i_2}\rangle, r_{i_3}\rangle \ldots, r_{i_k}\rangle$$

using every generator at most once. Let us take the lowest rank counterexample X and use reachability to get $X = \langle Y, r\rangle$. If r appears in the description of Y using every generator at most once, then $X = Y$ by absorption and hence X is not a counterexample. Otherwise $\langle Y, r\rangle$ is the required form of X.

Absorptivity is a natural constraint that basically requires the attributes of a merged record be formed as the union of the original values. Note that it is easy to give examples with no absorption, for example the one in Section 3.1 that always keeps the attribute of the most recently merged record only. It remains an open question to find the weakest possible definition of absorptivity for Theorem 1 to hold. The condition in equation (4) is insufficient since if we merge four records in two different order, their equality will not be guaranteed.

Reachability is a frequent requirement for set systems. It is easy to construct counterexamples when the confidence of two entities belonging to the same household reaches the threshold to unify them only after several duplicates of each entity are merged by correcting and filling their attributes. However our algorithm is not capable of solving the general problem with no reachability. Here additional, weaker constraints may be necessary to extend our method.

3.4 Entity Closure: Towards an Efficient Algorithm

Towards solving the general entity resolution problem with the constraints as in Theorem 1, we introduce the concept of entity closure by basically imposing an additional weak transitivity constraint on the match relation.

The **entity closure** for an entity resolution problem formulated by $\mathcal{E} = (2^R, \cup, \sim)$ is defined by $\mathcal{E}^* = (2^R, \cup, \sim^*)$ where $\sim^*$ is the transitive closure of $\sim$. The entity resolution for the closure is an upper bound for the solution of the original entity resolution problem and can be generated by computing the maximal connected components of the graph defined by $(R, \sim|_{R\times R})$.

Given the entity closure, the solution of the original entity resolution problem can be reached by post-processing, i.e. by separating larger entities. We will first compute the transitive closure in an efficient scalable distributed way that provides an upper bound for non-transitive cases for post-processing. The rationale of the algorithm is that in real data, most matches have minor number of

oarieli@mta.ac.il

non-transitive instances, but some of the entities in the closure have to be split, e.g. when an ambiguous record belongs to more than one entities. Computing by closure followed by post-processing can be done efficiently, provided the entities in the closure are small.

4 Features

The entity match relation can be defined in many ways. In its simplest form, we may match entities with identical attribute values. For example, two client entities can be merged if we find two identical social security numbers or id card numbers. In such a case, match between record sets can be generated from match between record pairs: If two records can be connected through pairwise matches, then these two records belong to the same entity. Two entities match, if they have a matching record pair.

As shown in the example of Fig. 1, we may not necessary want to merge entities with just a few equal attributes but instead assign confidence thresholds that may result in overlapping entities. In this case the match relation generated by the record pairs forms an upper bound of the original match relation as in Section 3.4.

Next we give a formal but practical model to express a match relation using features that suits our distributed algorithm well. We distinguish three aspects of a feature. These three aspects can be independently formulated, and can be built around domain knowledge.

First, we isolate properties of entities based on attributes of records (eg. the birth date and maiden name pair of a client). Let a **feature** be an $E \to X$ function f, where X is an arbitrary set. Elements of X represent discriminating and independent properties of an entity, and can be for example numbers, strings or any data structures (eg. arrays of strings).

Note that the value of a feature can be a set or a single value, e.g. the 'name' feature may keep both current name and maiden name of a person record if they differ, resulting a 2-element set of feature values; or we can define it as a single value by concatenation of the two strings using a separator; or we can keep only one of the two. The situation is similar for non-singleton entities arising by merging records. The definition of feature f can be extended simply as the set union: $f(x \cup y) = f(x) \cup f(y)$. Other feature definitions may involve minimum or maximum, e.g. $f(x \cup y) = min\{f(x), f(y)\}$.

Second, let a **feature-based match function** be an $X \times X \to \{\text{true}, \text{false}\}$ partial function, where X is a set of feature values. It must be reflexive and symmetric but not necessarily transitive. We denote the match function of a given f feature as $\sim_f$. The match function is defined only for the values the feature maps to. If we use multiple features, the unified feature-based match function is defined as a disjunction. For some feature set $f_1, ...f_k$ and entities $e, e' \in E$:

$$e \sim_{f_1,...,f_k} e' \Leftrightarrow e \sim_{f_1} e' \vee e \sim_{f_2} e' \vee ... \vee e \sim_{f_k} e'.$$

oarieli@mta.ac.il

Algorithm 1. Feature-based ER by Map-Reduce

input: Entity set E over a distributed file system.
output: $E' = ER(E)$

```
1: for all features f_i do
2:     sort all r ∈ E records by representative values rep_i(r)
3:     for all representative values rep_i(r) of r ∈ E do
4:         for all pairs of records r, r' with rep_i(r) = rep_i(r') do
5:             if f_i(r) ∼_i f_i(r') then
6:                 write (ID(r), ID(r')) to graph G
7: Map-Reduce connected components(G)
8: sort records by component ID and merge groups of identical ID
```

The third aspect of a feature is indexability. To ensure the efficient computation of feature-based matches, we expect features and their match functions to be indexable. Let the **feature mapping function** of a feature f be a partial function $rep_f : X \to 2^O$ with some ordered set O. The mapping produces representative elements used to index the entities, to produce match candidates.

The goal is to construct feature mappings that produce identical elements for matching entities by that particular feature. For **indexable features** a feature mapping can be constructed so that all entity e' matching a given e by $\sim_f$ can be found through the equality of representative values:

$$\forall e, e' \in E, e \sim_f e' \Rightarrow \exists o \in O : o \in rep_f(f(e)) \wedge o \in rep_f(f(e')).$$

Constructing indexable feature mappings is not always easy and may involve similarity indexes. In a similarity index, entities may have many representative values. For example, string similarity can be indexed by n-grams, or by the words they contain, based on the given similarity function.

5 Distributed Algorithm

Next we modify our existing distributed algorithm published in [37]. It is based on Hadoop [42], an open source implementation of the Map-Reduce framework [16]. The following version solves the entity closure problem based on indexable features.

The algorithm works in two rounds. The first round, Algorithm 1, iterates through all features. For each, it sorts attribute values and records all potential matches in a graph file. Then the connected component Algorithm 2 is called that assigns a component ID to all records. Finally, the last line of the main algorithm merges all records with the same ID. In this step, additional split heuristics can be implemented to undo some of the unnecessary merges if necessary.

In Algorithm 1 we assign IDs to records as follows. If there are entities that consist of more than one record at start, we split it into two records, both with the same ID.

oarieli@mta.ac.il

Algorithm 2. Map-Reduce connected components

input: Graph G of record IDs.
output: Component ID for all record IDs.

```
sort G to form sequences S_i = {i, ID_i, list of edges (i, j)}
change = true
while change = true do
    change = false
    Map:
    for all IDs i do
        for all IDs j with (i, j) ∈ S_i do
            emit ID_i to reducer j
        emit entire S_i to reducer i
    Reduce:
    for all reducers j do
        ID' = min of all ID values received
        if ID' < ID_j then
            change = true
            replace ID_j by ID in S_j
        write S_j
```

We describe the connected component Algorithm 2 in detail. The algorithm implements the matrix multiplication based all-pairs reachability algorithm of [15, Section 25] in a way similar to [25]. Two ingredients are the reduction of the problem to iterated matrix multiplication with a modified associative operation and the implementation of the matrix operation over Hadoop. For the first, let us replace addition by the minimum function and let

$$ID_j = min(ID_j, min_{i:(ij) is\ an\ edge}(ID_i)). \tag{5}$$

In iteration s, this method selects the minimum value in the s step neighborhood of every record. If we record the fact that some ID_j decrease in an iteration then we can terminate if there is no change.

Finally we show how to compute the matrix-vector multiplication type step of 5 by Map-Reduce. Starting at line 5, mapper i sends its current ID to reducer j for all edges ij in the graph to prepare the data needed to compute 5. In addition, reducer j starting in line 10 must write data S_j suitable for the next matrix-vector multiplication iteration. In addition to ID_j, this S_j must contain the edges out of record j. For this purpose, mapper i sends its entire data S_i to reducer i, completing the description of the algorithm.

The running time of the algorithm is $O(\ell(n \log n)/t)$ for ℓ features over t servers and $O(sn/t)$ for connected components over t servers where s is the size of the largest component.

oarieli@mta.ac.il

6 Examples of Indexable Features

Within the framework of features, a variety of match relations can be formulated, starting with the simplest equality based match towards more complex match relations of more than one attributes, similarity-based heuristics or probabilistic decisions. In the following we show how match relations can be developed for the given entities and business problem that enable efficient indexing and algorithm scalability.

Equality-based Match Functions. The simplest way to define features is to rely on attributes with equality. In this case attribute values can be used as representative values too, enabling efficient candidate generation: all candidate pairs match, and all matching pairs will be candidates.

Match Functions with Multiple Attributes. Combining multiple attributes to form a feature is an obvious next step. The simplest case is to concatenate the two attribute values, and then treat the concatenation as a simple attribute, used both for match functions and for feature mappings. Arbitrary features and match relations can be however defined freely on the attributes. The only restriction is that feature mappings have to be designed carefully not to miss potential record matches.

Note that sets of attributes may represent the same concept and hence form a single feature. For example, it is a common case to store multiple phone numbers for clients – a general phone number, a cell and a home phone. A common phone feature does not distinguish between these attributes, equality of any phone numbers may mean an entity match. Such attributes can be represented by and mapped to themselves, resulting in more than one representative element for one record.

Probabilistic Match Functions. ER tasks are always interpreted in an uncertain environment, even if we define exact rules. The main ER problem arises from the fact that observations of real world entities are erroneous and vague in some sense. Therefore, entity resolution based on exact rules is not flexible enough; probabilistic models for these uncertain statements are preferred.

Besides the plain database of entity records, a-priori knowledge may be available to improve ER quality. For example, we may have information on distributions of attribute values or cardinalities of entity groups. Similarity measures shall consider statistical properties of the given entity set: two records having the name "John Smith" match with lower probability than two having "Dunstan Everitt". For an other example, the phone number "+36 20 222 2222" is valid, and two records having these numbers can be matched. However, if we look at the frequency of this particular number in our database, it turns out that this number is outstandingly common. It is used when a phone number has to be given, but the client does not provide it. Using the frequency information these matches can be avoided. The external information should be incorporated into the ER model or represented as extra attributes.

oarieli@mta.ac.il

In the following we inspect events where we take two records, r_1 and r_2 of some hidden real-world entities e_1 and e_2. We think of record matching as assigning a probability to the hypothesis "r_1 and r_2 describe the same entity ($e_1 = e_2$)". We incorporate in our model external knowledge as prior probabilities, and expert knowledge as probability estimations.

For a given attribute a, let records r_1, r_2 take the values a_1 and a_2 – we denote these events as a_1 and a_2 for short. Our main goal is to provide a sound estimation for $P(e_1 = e_2 \mid a_1 \cap a_2)$. We expect events a_1 and a_2 to be independent. Using Bayes' formula,

$$P(e_1 = e_2 \mid a_1 \cap a_2) = \frac{P(a_1 \cap a_2 \mid e_1 = e_2)P(e_1 = e_2)}{P(a_1 \cap a_2)} = \\ = \frac{P(a_1 \cap a_2 \mid e_1 = e_2)P(e_1 = e_2)}{P(a_1)P(a_2)}.$$

In Algorithm 1 the estimated probability above is used to decide matches: All pairs above a given threshold are matching pairs.

$P(e_1 = e_2)$ is a constant value, and can be estimated by the help of domain knowledge. For example, if we are looking for clients in a database of client records, and the domain expert states that the company have n clients all having approximately the same number of records, then this probability is $1/n^2$.

We have prior knowledge for $P(a_1)$ and $P(a_2)$ too: a good estimation can be given using the distribution of attribute values. For example, the given name "John" is be more probable than "Everitt".

The most interesting part is $P(a_1 \cap a_2 \mid e_1 = e_2)$. That is, if we assume that the two records belong to the same real world entity, then what is the probability of having the given attribute values. For strict equality-based matching, we can say that this probability is zero if the attributes are not equal. More advanced heuristics can also be built, for example, it is a common case for postal addresses to build decision trees based on the parts of the address. Or, we can incorporate the probability of typos. If we have training data, then classifiers can also be trained to learn the probabilities.

Probabilistic Match Functions with Multiple Attributes. Probabilistic match can be extended to handle more than one attribute. Again, we take two records, r_1 and r_2 of some hidden e_1 and e_2 real-world entities. Now we use not only an a, but a b attribute with values b_1 and b_2 (with the same notion for the corresponding events). Using the Bayes' formula and supposing the independence of a and b events,

$$P(e_1 = e_2 \mid a_1 \cap a_2 \cap b_1 \cap b_2) = \frac{P(a_1 \cap a_2 \cap b_1 \cap b_2 \mid e_1 = e_2)P(e_1 = e_2)}{P(a_1 \cap a_2 \cap b_1 \cap b_2)} = \\ \frac{P(a_1 \cap a_2 \mid e_1 = e_2)P(b_1 \cap b_2 \mid e_1 = e_2)P(e_1 = e_2)}{P(a_1)P(a_2)P(b_1)P(b_2)}.$$

oarieli@mta.ac.il

Probabilistic Match Functions and Missing Values. We must consider the semantics of null ($\emptyset$) values to decide how they should be treated. In the probabilistic model we can incorporate semantic and domain knowledge about null values.

As an example, we consider two attributes as in probabilistic match with multiple attributes. It can be generalized to more attributes as well. Since the case is symmetric regarding a and b as well as r_1 and r_2, the following relevant cases can occur:

1. One attribute is null in both records: $a_1 \neq \emptyset, a_2 \neq \emptyset, b_1 = b_2 = \emptyset$
2. One attribute is null in only one of the records: $a_1 \neq \emptyset, a_2 \neq \emptyset, b_1 \neq \emptyset, b_2 = \emptyset$
3. Each attribute has a null in one of the records: $a_1 = \emptyset, a_2 \neq \emptyset, b_1 \neq \emptyset, b_2 = \emptyset$

Adapting the Bayes formula for the above cases can be based on the principle that the null value can mean any value and we have no observation (event) on that value. The most interesting Case 2 can for example be written as follows, assuming independence:

$$P(e_1 = e_2 \mid a_1 \cap a_2 \cap b_1) = \frac{P(a_1 \cap a_2 \cap b_1 \mid e_1 = e_2)P(e_1 = e_2)}{P(a_1 \cap a_2 \cap b_1)} = \frac{P(a_1 \cap a_2 \mid e_1 = e_2)P(b_1 \mid e_1 = e_2)P(e_1 = e_2)}{P(a_1)P(a_2)P(b_1)}.$$

If we take the strict equality case, i.e. the value b of the real-world entities e_1 and e_2 the data records refer to must agree if they are equal, then the probability of data record r_2 (with a null value on b_2) referring to the same entity as record r_1 is the probability of the real value b of entity e_2 behind the missing observation b_2 is the same as the observed b_1 value. Therefore, $P(b_1 \mid e_1 = e_2) = P(b_1)$ and the above formula becomes similar to the one-attribute case. This matches our common-sense assumption that having a value for an attribute b in record r_1 is irrelevant if the same attribute in r_2 is unknown.

Probabilistic Match Functions using Similarities. Estimation of $P(a_1 \cap a_2 \mid e_1 = e_2)$ in the probabilistic match model can be facilitated by similarity metrics between a_1 and a_2, for example, edit distance, Soundex or stemming and token similarities of a client name.

Without going into details, we note that it is not straightforward to construct a feature mapping for similarity-based probabilistic match functions. For edit distance, as the most popular choice in a wide spectrum of applications, n-gram indexing enables finding all matching pairs above a given similarity threshold. N-grams may result however in too many representative elements and inefficient indexing. As an other example, [45] defines mappings from string space to integer space to support similarity-based string search; these methods can be applied when building feature mappings.

7 Experiments

Experiments were performed on 15-server Linux farms containing identical dual core 3 GHz Pentium CPUs, 4 GB of main memory. Software versions were Sun

oarieli@mta.ac.il

Java 1.6 and Hadoop 0.20.3. We configured Hadoop to use all available internal memory. The largest data sets do not fit in the memory of one node but still fit in the entire 15×4 GB of the cluster.

The data set is provided by AEGON Hungary Insurance Ltd. containing approximately 13 million client records. Records consist of both personal attributes (names, birth data, tax number, etc.), internal identifiers, postal addresses and phone numbers. The data set can be considered as a snapshot of clients; the newest available attributes are contained for all records if the source system supports historical data. The size of the input with a rich set of attributes is around 3 GB in a flat CSV file.

We used random sampling to obtain smaller subsets. We also used selection heuristics to influence the count of records per client and per household. For example, selecting all records for the family name 'Smith' instead of a random sample will increase the match count. We created larger data sets by replication and random permutation. In each replica, we added a version tag to all attributes so that the original structure of matches was preserved by replica but no new matches were introduced between replicas. For edit-distance-based match we shifted the codes of characters in the strings by constant values.

7.1 Features and Candidate Generation

Personal features that enable finding records of the same people were used in all settings. Postal address and phone numbers were used only for household identification. We applied equality-based match on personal attributes or on attribute combinations, with the attributes themselves used for indexing in most cases.

The following features were used:

f_p	name & maiden name, birth date, mother name
f_s	social security number
f_x	tax number
f_{c1}, f_{c2}	source system-specific codes
f_t	phone number & mobile phone number
f_t^{b1}, f_t^{b2}	phone number(s) in two different probabilistic models
f_a	full postal address
f_a^f	postal address (index) with family name
f_{a1}	postal address up to house number
f_{a1}^f	postal address up to house number (index) with family name
f_{a2}	postal address up to street and postal code

Multi-attribute features are indexed by string concatenation of attributes. Probabilistic phone number features are for avoiding false merges by frequent dummy phone numbers. Combined features with postal address and family name are indexed by postal address. Since each record contains two postal address attributes, these are merged into one address-related feature, just as phone and mobile phone numbers into another.

oarieli@mta.ac.il

For experiments 1–9, we used the following feature combinations:

Features →	personal data					phone			address-related				
Case ↓	f_p	f_s	f_x	f_{c1}	f_{c2}	f_t	f_t^{b1}	f_t^{b2}	f_a	f_a^f	f_{a1}	f_{a1}^f	f_{a2}
1	•	•	•	•	•								
2	•	•	•	•	•	•							
3	•	•	•	•	•		•						
4	•	•	•	•	•			•					
5	•	•	•	•	•	•			•				
6	•	•	•	•	•	•				•			
7	•	•	•	•	•	•					•		
8	•	•	•	•	•	•						•	
9	•	•	•	•	•	•							•

The resolved entities correspond to persons (clients) in cases 1–4, and households in cases 5–9.

7.2 Scalability

Previous work that we are aware of assume only 10 to 100 thousand records as input. To give an example, in [27] a distributed algorithm is described with similarity-based match, tested on 114 thousand records. Algorithms of [39] were tested closest to our database size with 10 million records.

As also shown in [37], our MapReduce ER algorithm is scalable to hundred millions of records. Figure 2 demonstrates the running times on the client data set, solving the ER problem of clients. We applied the 1st feature combination of Section 7.1, not using the address and phone number attributes, and using strict attribute equality-based feature match.

Figure 2 also provides an inaccurate, but useful overview of several ER algorithms on client data, namely the following:

- Java-F-Swoosh: a Java implementation of the best previously known generic entity resolution algorithm (F-Swoosh [2]),
- DB-GER: our best relational entity resolution algorithm, based on a commercial relational database (see [35]),
- index-ER-BDB: our best efficient indexing algorithm built on Java and Berkeley DB (see [36]),
- MapReduce: our best distributed ER algorithm using 15 computer nodes and Hadoop implementation.

We implemented these conceptually different ER algorithms in more or less different environments, therefore the comparison is not entirely correct. Nevertheless, the superiority of the MapReduce algorithm is clearly visible.

oarieli@mta.ac.il

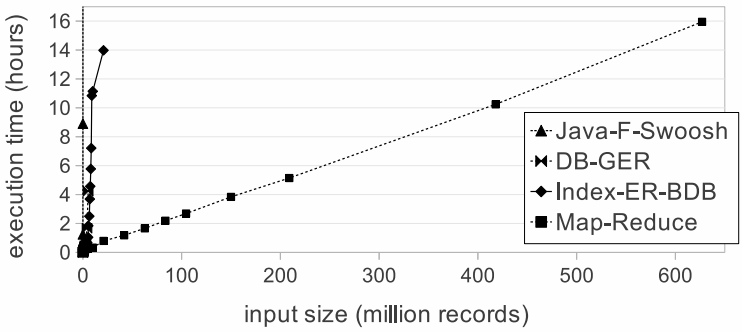


Fig. 2. Scalability of different entity resolution algorithms

7.3 Features and Entity Sizes

To find all households contained in the data set we used the different feature combinations of Section 7.1. Figure 3 shows how execution time increases with average entity size. The algorithm still seems to be useful when relatively large entities are generated, containing 5-6 records in average, which is much more than our motivating problem of households require.

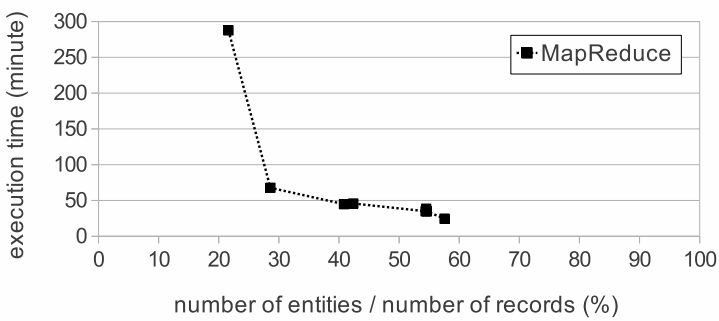


Fig. 3. Running time of the distributed algorithm with different entity sizes

Figure 4 shows the entity size distribution for one of the cases, when households are generated so that the last part of the address (the unit number inside a house) is omitted. This case corresponds the longest execution process shown in Fig. 3. The input contains 13.3M records and the total number of output entities is about 2.9M.

oarieli@mta.ac.il

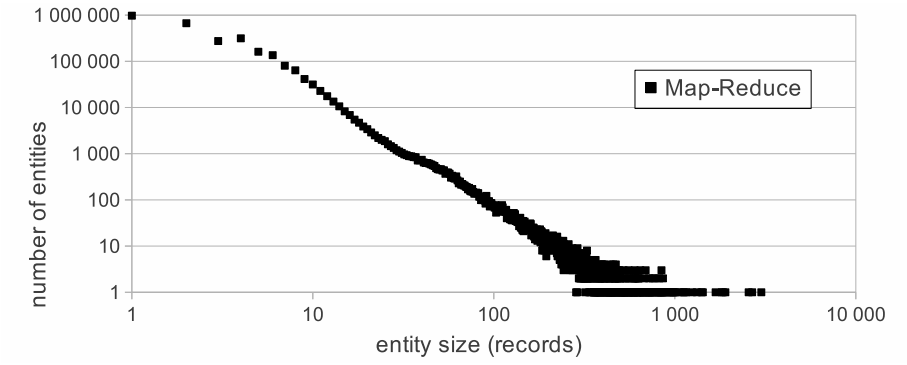


Fig. 4. Distribution of the number of entities and their sizes (households based on partial address)

8 Conclusions and Future Work

In this paper we introduced a generalized theoretical approach and the the accessible record subset model to formulate entity resolution problems in a flexible way. We gave a framework to define probabilistic and similarity-based match relations. We showed that resolution of large entities can be efficiently solved by a scalable distributed algorithm. We demonstrated the usability of our methods by identifying households of insurance client records.

There are several important areas of research to pursue. One issue is how to switch from similarities to probabilities. Several similarity metrics and indexing methods for similarity searches can be found in literature. The methods working efficiently with our distributed algorithm and also feasible for domain experts have yet to be found.

Another issue is the potential use of probability values. Overlapping or density-based clustering of records may describe the entities of the given business problem better. Therefore, it would be profitable to use the probability values generated when matching record pairs, as weights for the edges of the record graph.

Acknowledgments. To András Vereczki and Zoltán Hans as domain experts on the AEGON Hungary side for discussion on the problem formulation and clarification of the user requirements.

References

1. Benjelloun, O., Garcia-Molina, H., Gong, H., Kawai, H., Larson, T.E., Menestrina, D., Thavisomboon, S.: D-Swoosh: A family of algorithms for generic, distributed entity resolution. In: Proc. 27th Int. Conf. on Distributed Computing Systems (2007)

oarieli@mta.ac.il

2. Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S.E., Widom, J.: Swoosh: a generic approach to entity resolution. VLDB J. 18(1), 255–276 (2009)
3. Bhattacharya, I., Getoor, L.: A Latent dirichlet model for unsupervised entity resolution. In: SIAM International Conference on Data Mining, pp. 47–58 (2006)
4. Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. ACM Trans. Knowl. Discov. Data 1(1), 5 (2007)
5. Bhattacharya, I., Getoor, L., Licamele, L.: Query-time entity resolution. In: Proc. 12th ACM SIGKDD, pp. 529–534 (2006)
6. Bhattacharya, I., Godbole, S., Joshi, S.: Structured entity identification and document categorization: two tasks with one joint model. In: KDD 2008: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 25–33. ACM, New York (2008)
7. Bilenko, M., Mooney, R.: Adaptive duplicate detection using learnable string similarity measures. In: Proc. 9th ACM SIGKDD, pp. 39–48 (2003)
8. Boley, M., Horváth, T., Poigné, A., Wrobel, S.: Efficient Closed Pattern Mining in Strongly Accessible Set Systems (Extended Abstract). In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 382–389. Springer, Heidelberg (2007)
9. Chaudhuri, S., Sarma, A.D., Ganti, V., Kaushik, R.: Leveraging aggregate constraints for deduplication. In: SIGMOD 2007, pp. 437–448. ACM (2007)
10. Chen, S., Borthwick, A., Carvalho, V.R.: The case for cost-sensitive and easy-to-interpret models in industrial record linkage. In: 9th International Workshop on Quality in Databases (2011)
11. Christen, P.: Automatic record linkage using seeded nearest neighbour and support vector machine classification. In: KDD 2008, pp. 151–159. ACM (2008)
12. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. In: IEEE TKDE preprint (2011)
13. Christen, P., Churches, T., Hegland, M.: Febrl – A Parallel Open Source Data Linkage System. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 638–647. Springer, Heidelberg (2004)
14. Christen, P., Gayler, R., Hawking, D.: Similarity-aware indexing for real-time entity resolution. In: CIKM 2009, pp. 1565–1568. ACM (2009)
15. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press (2001)
16. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
17. Elmagarmid, A., Ipeirotis, P., Verykios, V.: Duplicate Record Detection: A Survey. IEEE TKDE, 1–16 (2007)
18. Fellegi, I., Sunter, A.: A theory for record linkage. Journal of the American Statistical Association 64(328), 1183–1210 (1969)
19. Getoor, L., Diehl, C.: Link mining: a survey. ACM SIGKDD Explorations Newsletter 7(2), 3–12 (2005)
20. Gravano, L., Ipeirotis, P., Jagadish, H., Koudas, N., Muthukrishnan, S., Srivastava, D.: Approximate string joins in a database (almost) for free. In: VLDB, pp. 491–500 (2001)
21. Guo, S., Dong, X.L., Srivastava, D., Zajac, R.: Record linkage with uniqueness constraints and erroneous values. Proc. VLDB Endow. 3, 417–428 (2010)
22. Hall, R., Sutton, C., McCallum, A.: Unsupervised deduplication using cross-field dependencies. In: KDD 2008, pp. 310–317. ACM (2008)

oarieli@mta.ac.il

23. Han, H., Xu, W., Zha, H., Giles, C.: A hierarchical naive Bayes mixture model for name disambiguation in author citations. In: Proc. 2005 ACM Symposium on Applied Computing, pp. 1065–1069 (2005)
24. Hernández, M., Stolfo, S.: Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery 2(1), 9–37 (1998)
25. Kang, U., Tsourakakis, C., Faloutsos, C.: Pegasus: A peta-scale graph mining system implementation and observations. In: ICDM, pp. 229–238. IEEE (2009)
26. Kim, H.-S., Lee, D.: Parallel linkage. In: CIKM 2007. ACM (2007)
27. Kirsten, T., Kolb, L., Hartung, M., Gross, A., Köpcke, H., Rahm, E.: Data partitioning for parallel entity matching. Computing Research Repository (2010)
28. Köpcke, H., Rahm, E.: Training selection for tuning entity matching. In: QDB/MUD, pp. 3–12 (2008)
29. Köpcke, H., Rahm, E.: Frameworks for entity matching: A comparison. Data Knowl. Eng. 69, 197–210 (2010)
30. Köpcke, H., Thor, A., Rahm, E.: Evaluation of entity resolution approaches on real-world match problems. Proc. VLDB Endow. 3, 484–493 (2010)
31. McCarthy, J., Lehnert, W.: Using decision trees for coreference resolution. In: Proc. 14th Int. Conf. on Artificial Intelligence, pp. 1050–1055 (1995)
32. Menestrina, D., Benjelloun, O., Garcia-Molina, H.: Generic entity resolution with data confidences. In: CleanDB Workshop, pp. 25–32 (2006)
33. Menestrina, D., Whang, S.E., Garcia-Molina, H.: Evaluating entity resolution results. Proc. VLDB Endow. 3, 208–219 (2010)
34. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: SIGKDD, pp. 269–278 (2002)
35. Sidló, C.I.: Generic Entity Resolution in Relational Databases. In: Grundspenkis, J., Morzy, T., Vossen, G. (eds.) ADBIS 2009. LNCS, vol. 5739, pp. 59–73. Springer, Heidelberg (2009)
36. Sidló, C.I.: Entity resolution with heavy indexing. In: Proc. ADBIS, CEUR Workshop Proceedings (2011)
37. Sidló, C.I., Garzó, A., Molnár, A., Benczúr, A.A.: Infrastructures and bounds for distributed entity resolution. In: 9th International Workshop on Quality in Databases (2011)
38. Talburt, J.R.: Entity Resolution and Information Quality, 1st edn. Morgan Kaufmann (2010)
39. Weis, M., Naumann, F., Jehle, U., Lufter, J., Schuster, H.: Industry-scale duplicate detection. Proc. of the VLDB Endow. 1(2), 1253–1264 (2008)
40. Whang, S.E., Garcia-Molina, H.: Entity resolution with evolving rules. Proc. VLDB Endow. 3, 1326–1337 (2010)
41. Whang, S.E., Menestrina, D., Koutrika, G., Theobald, M., Garcia-Molina, H.: Entity resolution with iterative blocking. In: Proc. 35th Int. Conf. on Management of Data, pp. 219–232. ACM (2009)
42. White, T.: Hadoop: The Definitive Guide. Yahoo Press (2010)
43. Wick, M.L., Rohanimanesh, K., Schultz, K., McCallum, A.: A unified approach for schema matching, coreference and canonicalization. In: KDD 2008, pp. 722–730. ACM (2008)
44. Yakout, M., Elmagarmid, A.K., Elmeleegy, H., Ouzzani, M., Qi, A.: Behavior based record linkage. Proc. VLDB Endow. 3, 439–448 (2010)
45. Zhang, Z., Hadjieleftheriou, M., Ooi, B.C., Srivastava, D.: Bed-tree: an all-purpose index structure for string similarity search based on edit distance. In: SIGMOD, pp. 915–926. ACM (2010)

oarieli@mta.ac.il

On Real-Valued Evaluation of Propositional Formulas

Aleksandar Perović[1], Dragan Doder[2], and Zoran Ognjanović[3]

[1] University of Belgrade, Faculty of Transportation and Traffic Engineering
pera@sf.bg.ac.rs
[2] University of Belgrade, Faculty of Mechanical Engineering
ddoder@mas.bg.ac.rs
[3] Mathematical Institute of Serbian Academy of Sciences and Arts
zorano@mi.sanu.ac.rs

Abstract. Arguably, $[0,1]$-valued evaluation of formulas is dominant form of representation of uncertainty, believes, preferences and so on despite some theoretical issues - most notable one is incompleteness of any unrestricted finitary formalization. We offer an infinitary propositional logic (formulas remain finite strings of symbols, but we use infinitary inference rules with countably many premises, primarily in order to address the incompleteness issue) which is expressible enough to capture finitely additive probabilistic evaluations, some special cases of truth functionality (evaluations in Lukasiewicz, product, Gödel and $Ł\Pi\frac{1}{2}$ logics) and the usual comparison of such evaluations. The main technical result is the proof of completeness theorem (every consistent set of formulas is satisfiable).

1 Introduction

One of possible ways to perceive any kind of propositional logic is to see it as a method (or art) of extending evaluations of propositional letters (variables) to evaluations of all formulas. Theoretical freedom aside, it would be prudent for the extension process to have some nice mathematical machinery preferably represented in common mathematical knowledge, which essentially boils down to elementary arithmetics, embodied in ordered fields $\langle \mathbb{Q}, +, \cdot, \leqslant 0, 1\rangle$ and $\langle \mathbb{R}, +, \cdot, \leqslant 0, 1\rangle$. Zadeh's seminal work on fuzzy sets [57] and Nilsson's work on probability logics and their application in expert systems [41] have launched $[0,1]$-valued logics as important and useful scientific tools in engineering community. Soon the rapid theoretical development has followed. We have tried to credit some of this huge amount of work throughout the paper. Though the list of references is rather extensive (54 entries), it is actually quite narrow both in size and scope.

There are two key issues with the $[0,1]$-valued propositional logics, both of them inherited from some well known facts about the ordered field of reals. The first of those is the noncompactness phenomenon, which is a consequence of the fact that the type of infinitely small element is omitted in $\langle \mathbb{R}, \leqslant, \{q \mid q \in \mathbb{Q}\}\rangle$. More precisely, if by $A(\alpha)$ we denote agent's estimation of α, then for any α

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 264–277, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

that is neither contradiction nor tautology, a nonrestricted system of evaluation assumes that $A(\alpha)$ can be any number from the real unit interval $[0, 1]$. If that is a case, than the following infinite list of conditions (logicians refer to such lists as theories)

$$A(\alpha) > 0.3,\ A(\alpha) < 0.31,\ A(\alpha) < 0.301,\ A(\alpha) < 0.3001, \ldots$$

is unsatisfiable ($A(\alpha)$ must be infinitesimally larger than 0.3, which is impossible in $\mathbb{R}$), in spite the fact that it is finitely satisfiable (any finite number of listed conditions can be satisfied). The other issue stems from the huge complexity of decision procedure for the first order theory of real closed fields (RCF). The last advance in this direction was Canny's PSPACE procedure for the existential fragment of the RCF [5].

Our aim is to address the first issue and provide complete axiomatization of significant part of reasoning about $[0, 1]$-valued evaluations of classical propositional formulas. The very basic idea behind any completion technique is to capture complete theories, i.e. theories that describe particular models. What is common to all $[0, 1]$-models is the nonexistence of infinitesimals. In other words, one must provide something like this:

If $A(\alpha)$ is infinitely close to a, then $A(\alpha)$ must be equal to A.

This property can be easily coded within the infinitary propositional logic L_{ω_1} (countable conjunctions and disjunctions are allowed) by the following formula:

$$\bigwedge_{n=0}^{\infty} a - 10^{-n} \leqslant A(\alpha) \leqslant a + 10^{-n} \ \rightarrow\ A(\alpha) = a.$$

Decidability requirement dictates finitary form of formulas. Fulfillment of this condition forbids usage of infinitary formulas. However, we can exercise a bit more freedom in deduction by means of infinitary inference rules and actually achieve the same effect. In other words, inference rule

From the set of premises $\{a - 10^{-n} \leqslant A(\alpha) \leqslant a + 10^{-n} \mid n \in \mathbb{N}\}$ infer $A(\alpha) = a$

has the same effect as the above infinitary formula.

The essence of several systems for $[0, 1]$-valued evaluations of formulas (probability logics, fuzzy logics, possibility logics) is that they can be expressed by Σ_0 RCF-sentences, provided that the initial language is enriched with the countably infinite set C of new symbols for constants. The elements of C actually represent desired evaluations. Here we present complete propositional axiomatization of that fragment.

The rest of the paper is organized as follows: in Section 2 we will develop our ground formalism, prove its completeness and decidability. Section 3 is devoted to applications, which are essentially rewriting of the standard axioms (for probability, possibility functions etc) in developed system. Concluding remarks are in the final section.

oarieli@mta.ac.il

2 Ground Formalism

In this section we will present what we call the ground (or the basic) formalism for reasoning about real valued evaluations of propositional formulas.

2.1 Syntax and Semantics

Let C be a countably infinite set such that $C \cap \mathbb{Q} = \emptyset$. The elements of C will be denoted by c and d, indexed if necessary. The set $Term$ of terms is inductively defined as follows:

- Any $c \in C$ is a term;
- For any $q \in \mathbb{Q}$, $\underline{q}$ is a term;
- If f and g are terms, then $(f \oplus g)$, $(f \odot g)$ and $(\ominus f)$ are also terms;
- Each term can be obtained by finite application of previous items.

In order to simplify notation we will use the standard convention for the omission of parentheses: the outermost parentheses would not be displayed, $\odot$ has greater priority than $\oplus$, $f \ominus g$ is an abbreviation for $f \oplus (\ominus g)$ and so on. Terms will be denoted by f, g and h, indexed if necessary.

Atomic formulas are formulas of the form $f = g$ and $f \preceq g$. The set For of formulas is defined as the set of all Boolean combinations of atomic formulas. Formulas will be denoted by ϕ, ψ and θ, indexed if necessary. Abbreviations such as $f \prec g$ or $f \succeq g$ have their usual meaning: $f \preceq g \wedge \neg(g \preceq f)$ and $g \preceq f$.

A model is any structure of the form $M = \langle \mathbb{R}, +, \cdot, -, \leqslant, \{c^M \mid c \in C\}, \{q \mid q \in \mathbb{Q}\}\rangle$, where $+$, $\cdot$, $-$ and $\leqslant$ are the usual addition, subtraction, multiplication and ordering of reals, while each c^M is a real number.

The value f^M of a term f in a model M is inductively defined as follows:

- The value of c in M is defined by c^M;
- $\underline{q}^M = q$;
- $(f \oplus g)^M = f^M + g^M$;
- $(f \odot g)^M = f^M \cdot g^M$;
- $(\ominus f)^M = -f^M$.

The satisfiability relation $\models$ is inductively defined as follows:

- $M \models f = g$ iff $f^M = g^M$;
- $M \models f \preceq g$ iff $f^M \leqslant g^M$;
- $M \models \neg\phi$ iff $M \not\models \phi$;
- $M \models \phi \wedge \psi$ iff $M \models \phi$ and $M \models \psi$.

As it is usual, sets of formulas are called theories. The notions of satisfiability, validity, finite satisfiability and semantical consequence are defined in the usual way.

Notice that compactness theorem fails for our system. Indeed, theory

$$T = \{c \succ \underline{0}\} \cup \{c \preceq \underline{10^{-n}} \mid n \in \mathbb{N}\}$$

oarieli@mta.ac.il

is both unsatisfiable and finitely satisfiable. Similarly, the theory

$$\Gamma = \{\underline{n} \preceq c \mid n \in \mathbb{N}\}$$

is also both unsatisfiable and finitely satisfiable.

2.2 Axioms and Inference Rules

We have divided axioms in the following groups:

Propositional axioms

A1. Substitutional instances of tautologies;

Equality and ordering axioms

A2. $f = f$;
A3. $f = g \to (\phi(\ldots, f, \ldots) \to \phi(\ldots, g, \ldots))$;
A4. $f \preceq f$;
A5. $f \preceq g \vee g \preceq f$;
A6. $f \preceq g \wedge g \preceq f \to f = g$;
A7. $f \preceq g \wedge g \preceq h \to f \preceq h$;
A8. $f \preceq g \to f \oplus h \preceq g \oplus h$;
A9. $f \preceq g \wedge \underline{0} \prec h \to f \odot h \preceq g \odot h$;

Bookkeeping axioms

A10. $\underline{q} \preceq \underline{r}$, if $q \leqslant r$;
A11. $\underline{q} \prec \underline{r}$, if $q < r$;
A12. $\underline{q} \oplus \underline{r} = \underline{q + r}$;
A13. $\underline{q} \odot \underline{r} = \underline{q \cdot r}$;
A14. $\ominus \underline{q} = \underline{-q}$;

Algebraic axioms

A15. $f \oplus g = g \oplus f$;
A16. $f \oplus (g \oplus h) = (f \oplus g) \oplus h$;
A17. $f \oplus \underline{0} = f$;
A18. $f \ominus f = \underline{0}$;
A19. $f \odot g = g \odot f$;
A20. $f \odot (g \odot h) = (f \odot g) \odot h$;
A21. $f \odot \underline{1} = f$;
A22. $f \odot (g \oplus h) = (f \odot g) \oplus (f \odot h)$;

Boundedness axioms

A23. $\underline{0} \preceq c \preceq \underline{1}$, $c \in C$.

oarieli@mta.ac.il

Before we state the inference rules let us briefly comment introduced axioms. Propositional axioms provide validity of tautology instances. Equality and ordering axioms provide the usual behavior of $=$ (congruence compatible with relations) and to ensure that $\preceq$ behaves like $\leqslant$. Bookkeeping axioms preserve the structure $\langle \mathbb{Q}, +, \cdot, -, \leqslant \rangle$. Algebraic axioms ensure that $\oplus$, $\odot$ and $\ominus$ actually behave like their semantical counterparts $+$, $\cdot$ and $-$. Finally, boundedness axioms are introduced primarily in order to avoid introduction of another inifinitary inference rule that would address the omission of the type of the infinitely large number in $\mathbb{R}$. The reader that is not familiar with the basic model theory can found necessary information on realizing and omitting types in [38].

Beside modus ponens, we have additional infinitary inference rule (so called Archimedean rule) that guaranties Archimedean models, i.e. that will render inconsistent any finitely satisfiable theory that propagates existence of proper infinitesimals. The explicit form of mentioned inference rules is given below.

R1. From ϕ and $\phi \to \psi$ infer ψ;
R2. From the set of premises $\{\phi \to \underline{q - 10^{-n}} \preceq f \mid n \in \mathbb{N}\}$ infer $\phi \to \underline{q} \preceq f$.

Intuitively, Archimedean rule R2 says that if a term f is infinitely close to a rational number q, than it must be equal to q. Consequently, any finitely satisfiable theory that propagates existence of proper infinitesimals becomes inconsistent in the introduced system. Implicative form in both premises and conclusions in R2 is technical detail that enables an easy proof of the deduction theorem.

Due to the presence of infinitary inference rules, the notion of a proof is somewhat different than in the classical case. The only difference is the length of inference, which by definition can be any countable successor ordinal. Regardless, ordinals are well ordered sets, so induction on the length of inference can be used quite similarly as in the classical case. More on ordinals can be found in graduate texts in model theory and set theory, see for instance [1, 29, 38].

It is easy to see that the following variant of the Archimedean rule is derivable:

From the set of premises $\{\phi \to f \preceq \underline{q + 10^{-n}} \mid n \in \mathbb{N}\}$ infer $\phi \to f \preceq \underline{q}$;

2.3 Completeness Theorem

Completion technique that we are going to present here is essentially the same technique that we have developed in order to overcome the well known non-compactness phenomenon in real valued probability logics, see [7, 10, 42, 46–50, 53, 54]. As it is usual in logic, the soundness part of the completeness theorem ($T \vdash \phi$ implies $T \models \phi$) can be proved using straightforward induction on the length of inference. Using (almost) the same argument as in the classical case (again the induction on the length of inference), we can prove the deduction theorem ($T, \phi \vdash \psi$ iff $T \vdash \phi \to \psi$). For instance the induction step that corresponds to the rule R2 goes as follows: suppose that $T, \phi \vdash \psi_n$ iff $T \vdash \phi \to \psi_n$ for all $n \in \mathbb{N}$, where ψ_n is $\psi \to \underline{q - 10^{-n}} \preceq f$. Since $\vdash (\phi \to (\psi \to \underline{q - 10^{-n}} \preceq f)) \leftrightarrow (\phi \wedge \psi \to \underline{q - 10^{-n}} \preceq f)$ (instance of tautology $(\alpha \to (\beta \to \gamma)) \leftrightarrow (\alpha \wedge \beta \to \gamma)$), we have that $T \vdash \phi \wedge \psi \to \underline{q - 10^{-n}} \preceq f$ for all $n \in \mathbb{N}$. By R4, $T \vdash \phi \wedge \psi \to \underline{q - 10^{-n}} \preceq f$, so $T \vdash \phi \to (\psi \to \underline{q - 10^{-n}} \preceq f)$.

oarieli@mta.ac.il

Lemma 1. *Suppose that T is a consistent theory and that $T, \underline{q} \preceq f$ is inconsistent. Then, there is $m \in N$ such that $T, f \prec \underline{q - 10^{-m}}$ is consistent.*

Proof. Contrary to the statement of the lemma, suppose that $T, f \prec \underline{q - 10^{-m}}$ is inconsistent for all $m \in \mathbb{N}$. Then, $T \vdash \underline{q - 10^{-m}} \preceq f$ for all $m \in \mathbb{N}$. If ϕ is any tautology instance, then $T \vdash phi \rightarrow \underline{q - 10^{-m}} \preceq f$ for all $m \in \mathbb{N}$. By Archimedean rule, $T \vdash \phi \rightarrow \underline{q} \preceq f$. Since $T \vdash \phi$, we have that $T \vdash \underline{q} \preceq f$. On the other hand, $T, \underline{q} \preceq f$ is inconsistent, so T must be inconsistent - contradiction with the presupposed consistency of T. □

Theorem 1. *Every consistent theory is satisfiable.*

Proof. The proof is divided in three parts:

1. Construction of the completion T^* of T;
2. Construction of the canonical model M^*;
3. Proof of the fact that $T^* \vdash \phi$ iff $M^* \models \phi$.

In pseudo-code the completion procedure can be described as follows:

$T^* := T$;
While $For \neq \emptyset$ do:
  Pick an arbitrary $\phi \in For$;
    If T^*, ϕ is consistent, then $T^* := T^* \cup \{\phi\}$;
    If T^*, ϕ is inconsistent and $\phi \neq \psi \rightarrow \underline{q} \preceq f$, then $T^* := T^* \cup \{\neg\phi\}$;
    If If T^*, ϕ is inconsistent and $\phi = \psi \rightarrow \underline{q} \preceq f$, then $T^* := T^* \cup \{\neg(\psi \rightarrow \underline{q} \preceq f), f \prec \underline{q - 10^{-m}}\}$, where m is the smallest positive integer such that T^* is consistent;
  $For := For \setminus \{\phi\}$;
End.

First of all, notice that just described completion technique differs from the classical one - it has an additional step that deals with the current formula that can be derived by means of the Archimedean rule. Here we have used a general logical technique which can be describes as follows:

> If the current formula ϕ is inconsistent with the current completion T^* and if ϕ can be derived by some inference rule R, then add to T^* both $\neg\phi$ and $\neg\psi$, where ψ is a premise in R needed for derivation of ϕ such that $T^* \cup \{\neg\phi, \neg\psi\}$ is consistent.

This technique is crucial for the proof of consistency of T^*. Indeed, if $T = \{c \succ \underline{0}\}$, then it is inconsistent with the formula $\phi : \ c = c \rightarrow c \preceq \underline{0}$. If we ignore the fact that ϕ can be derived from the set of premises $\{c = c \rightarrow c \preceq \underline{10^{-n}} \mid n \in \omega\}$ by Archimedean rule and just add $\neg\phi$ to the current value of T^*, then we can easily end up with inconsistent T^* since it is consistent with all formulas of the form $c = c \rightarrow c \preceq \underline{10^{-n}}$, so they can be all included in T^* after ω steps.

oarieli@mta.ac.il

Also notice that the existence of the m in the construction of T^* is provided by Lemma 1. It is useful to enumerate stages of the construction of T^*. Since For is a countable set, there are $\aleph_0$ such stages. We will refer to them as T_n. Using the argument similar to the one presented in [49, 50] we can show that T^* is deductively closed. Since each T_n is consistent, T^* must be consistent as well.

The canonical model M^* is defined as follows:

$$c^{M^*} = \sup\{q \in \mathbb{Q} \mid T^* \vdash \underline{q} \preceq c\},\ c \in C.$$

By induction on the complexity of a term it can be shown that

$$f^{M^*} = \sup\{q \in \mathbb{Q} \mid T^* \vdash \underline{q} \preceq f\}.$$

For instance, let $f^{M^*} = a$ and $g^{M^*} = b$. Suppose that $a > 0$, $b > 0$. Let $0 < q_{1,n} < a < q_{2,n}$, $0 < r_{1,n} < b < r_{2,n}$, $\lim\limits_{n\to\infty} q_{1,n} = \lim\limits_{n\to\infty} q_{2,n} = a$ and $\lim\limits_{n\to\infty} r_{1,n} = \lim\limits_{n\to\infty} r_{2,n} = b$. Then,

$$T^* \vdash \underline{q_{1,n} r_{1,n}} \preceq f \odot g \preceq \underline{q_{2,n} r_{2,n}},$$

so $ab = \sup\{q \in \mathbb{Q} \mid T^* \vdash \underline{q} \preceq f \odot g\}$, i.e. $(f \odot g)^{M^*} = ab$. The other cases (such as $a < 0$, $b > 0$) can be handled similarly.

As we have said earlier, the final step in the proof of the completeness theorem is the proof of the fact that

$$M^* \models \phi \ \text{ iff } \ T^* \vdash \phi.$$

The only difference with the classical case is in the proof of the statement for atomic formulas. Since $=$ is definable from $\preceq$, without loss of generality we can restrict ourselves to the formulas of the form $f \preceq g$. Furthermore, $\vdash f \preceq 0 \leftrightarrow \underline{0} \preceq g \ominus f$, so it is sufficient to consider only atomic formulas of the form $\underline{0} \preceq f$.

Suppose that $T^* \vdash \underline{0} \preceq f$. Let $a = \sup\{q \in \mathbb{Q} \mid T^* \vdash \underline{q} \preceq f\}$. Clearly, $0 \leqslant a$, so $M^* \models \underline{0} \preceq f$. Conversely, let $M^* \models \underline{0} \preceq f$. Then,

$$0 \leqslant f^{M^*} = \{q \in \mathbb{Q} \mid T^* \vdash \underline{q} \preceq f\}.$$

If $f^{M^*} > 0$, then clearly $T^* \vdash \underline{0} \preceq f$. Finally, if $f^{M^*} = 0$, then $T^* \vdash \underline{10^{-n}} \preceq f$, so by Archimedean rule $T^* \vdash \underline{0} \preceq f$. □

2.4 Decidability

Up to equivalence, each formula can be seen as a finite disjunction of systems of polynomial inequalities. Using the decidability result for polynomial weight formulas presented in [16] which is based on the PSPACE containment for decision procedure for the existential fragment of the first order theory of reals proved in [5], we can conclude that the satisfiability of introduced formulas is decidable and that complexity containment is PSPACE.

oarieli@mta.ac.il

3 Examples

Applications of previously developed formalism are based on rewriting of elements of C and adding new axioms for them. In this section we will show how to model simple probabilities, possibilistic distributions and evaluations in Gödel's fuzzy logic.

3.1 Simple Probabilities

Representation of uncertainty via probability functions on classical propositional formulas (or rather sets of models of such formulas) is one of the most important and natural concepts in reasoning under uncertainty. Some of books and papers that are relevant for our previous and current work are [15, 16, 21–23, 27, 30, 33–36, 40, 41, 43–45, 52, 55, 56].

In the single agent case we will assume that $\{P(\alpha) \mid \alpha \in For_{Cl}\} \subset C$, where For_{Cl} is the set of classical propositional formulas. Now we can assure that P actually behaves like probability by the following theory of the basic formalism:

- $P(\alpha) = 1$, whenever α is a tautology;
- $P(\alpha) = P(\beta)$, whenever α and β are equivalent;
- $P(\alpha \vee \beta) = P(\alpha) \oplus P(\beta) \ominus P(\alpha \wedge \beta)$;
- $P(\neg\alpha) = \underline{1} \ominus P(\alpha)$.

In the multi agent case the main idea is to use indices for designation of probability estimations for each agent. We believe it would be instructive to illustrate this in the formalization of the well known Grabisch's classification example treated in [20, 50]. Originally Grabisch has considered the classification of students according to their academic scores in three courses. As a preprocessing, academic scores should be normalized in the usual way: if we have n grades displayed increasingly as $a_1, a_2, \ldots, a_n$, than their normalization is defined by $\|a_i\| = \frac{i}{n}$. Courses C_1 C_2 and C_3 can be represented by propositional letters p_1, p_2 and p_3 respectively. So, normalized academic scores of the each student generate evaluation of propositional letters p_1, p_2 and p_3. As we have shown in [50], the obvious classification can be derived if we extend those evaluations on entire For_C by means of probability. Namely, if A is a nonempty finite set of pairwise disjoint propositional letters, then let $e\left(\bigwedge_{p\in A} p\right) = \prod_{p\in A} e(p)$. Here $e(p_i)$ coincides with normalized academic score in i-th course of some student and $e(p) = 1$ for $p \notin \{p_1, p_2, p_3\}$. What follows is a slight generalization of the original Grabisch's example.

Objects A, B, C and D are described by quality attributes p_1, p_2 and p_3, whose values are given in the following table:

object	p_1	p_2	p_3
A	0.75	0.9	0.3
B	0.75	0.8	0.4
C	0.3	0.65	0.1
D	0.3	0.55	0.2

oarieli@mta.ac.il

Objects A, B, C and D should be classified according to the following criteria:

ϕ_1: The average value of quality attributes;
ϕ_2: If the analyzed object is good with respect to p_1, then p_3 is more important than p_2. Otherwise, p_2 is more important than p_3.

The second classification criterion can be propositionally coded by the formula $(p_1 \wedge p_3) \vee (\neg p_1 \wedge p_2)$. In order to formally express just described problem, we will assume that $P_i(\alpha)$ is element of C for any classical propositional formula α. $i = 1, 2, 3, 4$. For each P_i we will add previously described probability axioms. Furthermore, we will assume that p_1, p_2 and p_3 are independent. In terms of probability this transcribes as follows:

- $P_i(p_1 \wedge p_2) = P_i(p_1) \odot P_i(p_2)$;
- $P_i(p_1 \wedge p_3) = P_i(p_1) \odot P_i(p_3)$;
- $P_i(p_2 \wedge p_3) = P_i(p_2) \odot P_i(p_3)$;
- $P_i(p_1 \wedge p_2 \wedge p_3) = P_i(p_1) \odot P_i(p_2) \odot P_i(p_3)$.

Moreover, we will formalize the given table in the obvious way:

- $P_i(p_j) = \underline{q_{i,j}}$ (for instance $q_{1,1} = 0.75$).

Now for each i we can formally compute

$$\underline{1/6} \odot (P_i(p_1) \oplus P_i(p_2) \oplus P_i(p_3)) \oplus \underline{1/2} P_i((p_1 \wedge p3) \vee (\neg p_1 \wedge p_2)),$$

formally compare calculated values and obtain desired classification.

3.2 Possibility and Necessity Functions

Possibility theory is one of the prominent branches of research in the field of nonmonotonic reasoning and reasoning under uncertainty, see [2, 4, 11–13, 17, 51]. Qualitative possibilities were introduced independently by Lewis in [32] and Dubois in [11], while qualitative necessities were introduced by Dubois in [11]. Though possibility and necessity relations are originally defined on algebras of sets, it is quite natural to treat them as relations on propositional formulas since all Boolean algebras are, up to isomorphism, Lindenbaum–Tarski propositional algebras.

A binary relation $\leqslant_\Pi$ on the set For_C of classical propositional formulas is a qualitative possibility if it satisfies the following conditions:

- $\leqslant_\Pi$ is a nontrivial weak order, i.e. it is linear (each pair of formulas is comparable), transitive and $\bot <_\Pi \top$, where $\bot$ is any contradiction, $\top$ is any tautology and $<_\Pi$ has the obvious meaning $\alpha \leqslant_\Pi \beta$ and $\beta \nleqslant_\Pi \alpha$;
- $\leqslant_\Pi$ is compatible with the equivalence of classical propositional formulas. More precisely, if α is equivalent to α' and β is equivalent to β', then $\alpha \leqslant_\Pi \beta$ iff $\alpha' \leqslant_\Pi \beta'$;
- For all $\alpha \in For_C$, $\bot \leqslant_\Pi \alpha$;

oarieli@mta.ac.il

- $\leqslant_{\Pi}$ satisfies so called disjunctive stability: if $\beta \leqslant_{\Pi} \gamma$, then, for all α, $\alpha \vee \beta \leqslant_{\Pi} \alpha \vee \gamma$.

A binary relation $\leqslant_N$ on For_C is a qualitative possibility if it has the following properties:

- $\leqslant_N$ is a nontrivial weak order;
- $\leqslant_N$ is compatible with the equivalence of classical propositional formulas;
- For all $\alpha \in For_C$, $\alpha \leqslant_N \top$;
- $\leqslant_N$ satisfies so called conjunctive stability: if $\beta \leqslant_{\Pi} \gamma$, then, for all α, $\alpha \wedge \beta \leqslant_{\Pi} \alpha \wedge \gamma$.

It is quite natural to treat qualitative possibilities and necessities as dual notions, since any qualitative possibility relation $\leqslant_{\Pi}$ generates the unique qualitative necessity relation $\leqslant_{N_{\Pi}}$ by

$$\alpha \leqslant_{N_{\Pi}} \beta \quad \text{iff} \quad \neg\beta \leqslant_{\Pi} \neg\alpha,$$

and vice versa. Moreover, for any possibility (necessity) relation $\leqslant_R$, the quotient structure $\langle For_{C/\sim}, \leqslant_{R/\sim}\rangle$ ($\alpha \sim \beta$ iff $\alpha \leqslant \beta$ and $\beta \leqslant \alpha$) is at most countable (For_C is countable) linear ordering with endpoints, so it can be embedded into $\langle [0,1], \leqslant\rangle$. If F is one such embedding, then by $f : \alpha \mapsto F(\alpha_{/\sim})$ is defined a distribution f of $\leqslant_R$, i.e. $\alpha \leqslant_R \beta$ iff $f(\alpha) \leqslant f(\beta)$. Such functions are also called possibility (necessity) functions.

If $\leqslant_{\Pi}$ is a qualitative possibility relation and π its arbitrary distribution, then

$$\pi(\alpha \vee \beta) = \max(\pi(\alpha), \pi(\beta)) \quad \text{(maxitivity condition)}.$$

Moreover, if $f : For_C \longrightarrow [0,1]$ satisfies maxitivity condition, then the relation $\leqslant_f$ defined by

$$\alpha \leqslant_f \beta \quad \text{iff} \quad f(\alpha) \leqslant f(\beta)$$

is a qualitative possibility relation. Dually, minitivity condition

$$\nu(\alpha \wedge \beta) = \min(\nu(\alpha), \nu(\beta))$$

fully characterizes qualitative necessity relations.

Hence, in order to formally capture evaluations that behaves like possibility functions and necessity functions, we just need to provide fulfillment of the corresponding characterizing condition (maxitivity for possibility functions, minitivity for necessity functions). In the single agent case we will assume that $\Pi(\alpha) \in C$ and $N(\alpha) \in C$ for any classical propositional formula α. Here $\Pi(\alpha)$ stands for the possibility measure of α, while $N(\alpha)$ stands for the necessity measure of α. We can obtain desired properties of Π and N by the following theory in the ground formalism:

- $\Pi(\alpha) = \Pi(\beta)$, if α and β are equivalent;
- $\Pi(\bot) \prec \Pi(\top)$;
- $\Pi(\alpha) \preceq \Pi(\beta) \rightarrow \Pi(\alpha \vee \beta) = \Pi(\beta)$;
- $N(\alpha) = N(\beta)$, if α and β are equivalent;
- $N(\bot) \prec N(\top)$;
- $N(\alpha) \preceq N(\beta) \rightarrow N(\alpha \wedge \beta) = N(\alpha)$.

oarieli@mta.ac.il

3.3 Evaluations in Gödel's Fuzzy Logic

Roughly speaking, Gödel's fuzzy logic is an extension of the basic logic (BL) with axioms that ensures that conjunction and disjunction behaves like min and max respectively. Furthermore, Gödel's implication (residuum of min) and negation are defined by

$$x \Rightarrow y = \begin{cases} 1 \ , \ x \leqslant y \\ y \ , \ y < x \end{cases}$$

and

$$\neg x = \begin{cases} 0 \ , \ x > 0 \\ 1 \ , \ x = 0 \end{cases} .$$

More information on fuzzy logic the reader can found in [14, 18, 21, 23, 24, 36, 37]. In order to formally capture semantics of Gödel' s fuzzy logic in the ground formalism (again in the single agent case), we will assume that $G(\alpha) \in C$ for any classical propositional formula C and introduce the following additional axioms:

- $\underline{0} \prec G(\alpha) \rightarrow G(\neg\alpha) = \underline{0}$;
- $G(\alpha) = \underline{0} \rightarrow G(\neg\alpha) = \underline{1}$;
- $G(\alpha) \preceq G(\beta) \rightarrow G(\alpha \rightarrow \beta) = \underline{1}$;
- $G(\alpha) \succ G(\beta) \rightarrow G(\alpha \rightarrow \beta) = G(\beta)$;
- $G(\alpha) \preceq G(\beta) \rightarrow G(\alpha \wedge \beta) = G(\alpha)$;
- $G(\alpha) \preceq G(\beta) \rightarrow G(\alpha \vee \beta) = G(\beta)$.

Since $G(\alpha) \preceq G(\beta) \vee G(\beta) \preceq G(\alpha)$ is a theorem of the basic formalism, the fifth axiom provides that in any model M, $G^M(\alpha \wedge \beta) = \min(G^M(\alpha), G^M(\beta))$. Similarly, the last axiom provides that $G^M(\alpha \vee \beta) = \max(G^M(\alpha), G^M(\beta))$. The first two axioms ensures that G^M behaves according to the truth table for Gödel's negation, while the third and the fourth axiom do the same for Gödel's implication. In this way we have ensured that, in any model M, $e(\alpha) = G^M(\alpha)$ is a correctly defined evaluation in Gödel's logic.

4 Conclusion

In [26] Halpern has shown that the first order theory of the ordered field of reals is one of natural candidates for the formal environment for studying uncertainty. However, complexity issue (EXPSPACE) can be seen as a prime motive for still ongoing research in this field. Σ_0-fragment of this theory (endowed with countably many new symbols for constants) is often enough to express various concepts such as simple and conditional probabilities, possibility and necessity functions, evaluations in Łukasiewicz, product,Gödel and Ł$\Pi\frac{1}{2}$ fuzzy logics. Hylbert style formal system presented here actually provides propositional axiomatization of mentioned fragment. Though the notion of the inference is changed due to the presence of the infinitary Archimedean inference rule, the general complexity is reduced to PSPACE and the strong completeness is achieved.

oarieli@mta.ac.il

Though we have not elaborated it, the concept of big-stepped probabilities can be used for the representation of nonmonotonic consequence relations in the standard real valued framework, provided that the set of propositional letters is finite. Using research presented in [10] we can represent big-stepped probabilities in the formalism introduced here, so some of the important concepts involving nonmonotonic reasoning presented in [3, 6, 9, 19, 28, 31, 39, 58, 59] can be coded here as well.

As it was shown in [25], temporal operators can be used to naturally represent the concept of evidence. Along the lines of research presented in [8], system presented here can be enriched with temporal operators so that its key properties (strong completeness and PSPACE complexity containment) would be preserved.

Acknowledgements. The work presented here was partially supported by the Serbian Ministry of Education and Science (projects ON174026, III44006 and III41013).

References

1. Barwise, J.: Admissible sets and structures: an approach to definability theory. Springer, Heidelberg (1975)
2. Benferhat, S., Dubois, D., Prade, H.: Possibilistic and standard probabilistic semantics of conditional knowledge bases. Journal of Logic and Computation 9(6), 873–895 (1999)
3. Bezzazi, H., Makinson, D., Pino Pérez, R.: Beyond rational monotony: some strong non-Horn rules for nonmonotonic inference relations. Journal of Logic and Computation 7(5), 605–631 (1997)
4. Boutilier, C.: Modal logics for qualitative possibility theory. International J. Approx. Reason. 10, 173–201 (1994)
5. Canny, J.: Some algebraic and geometric computations in PSPACE. In: Proc. of XX ACM Symposium on Theory of Computing, pp. 460–467 (1987)
6. Doder, D., Rašković, M., Marković, Z., Ognjanović, Z.: Measures of inconsistency and defaults. International Journal of Approximate Reasoning 51, 832–845 (2010)
7. Doder, D., Marinković, B., Maksimović, P., Perović, A.: A logic with conditional probability operators. Publications de l'Institut Mathématique 87(101), 85–96 (2010)
8. Doder, D., Marković, Z., Ognjanović, Z., Perović, A., Rašković, M.: A Probabilistic Temporal Logic That Can Model Reasoning about Evidence. In: Link, S., Prade, H. (eds.) FoIKS 2010. LNCS, vol. 5956, pp. 9–24. Springer, Heidelberg (2010)
9. Doder, D., Perović, A., Ognjanović, Z.: Probabilistic Approach to Nonmonotonic Consequence Relations. In: Liu, W. (ed.) ECSQARU 2011. LNCS, vol. 6717, pp. 459–471. Springer, Heidelberg (2011)
10. Doder, D.: A logic with big-stepped probabilities that can model nonmonotonic reasoning of system P. Publications de l'Institut Mathématique 90(104), 13–22 (2011)
11. Dubois, D.: Belief structures, possibility theory and decomposable confidence relations on finite sets. Comput. Artificial Intelligence 5(5), 403–416 (1986)
12. Dubois, D., Prade, H.: Conditional objects, possibility theory and default rules. In: Crocco, G., et al. (eds.) Conditionals: From Philosophy to Computer Science, pp. 311–346. Oxford University Press (1995)

oarieli@mta.ac.il

13. Dubois, D., Prade, H.: Qualitative possibility functions and integrals. In: Pap, E. (ed.) Handbook of Measure Theory, pp. 1499–1522. North-Holland (2002)
14. Esteva, F., Godo, L., Montagna, F.: The ŁΠ and Ł$\Pi\frac{1}{2}$ logics: two complete fuzzy logics joining Łukasiewicz and product logic. Archive for Mathematical Logic 40, 39–67 (2001)
15. Fagin, R., Halpern, J.: Reasoning about knowledge and probability. Journal of the ACM 41(2), 340–367 (1994)
16. Fagin, R., Halpern, J., Megiddo, N.: A logic for reasoning about probabilities. Information and Computation 87(1-2), 78–128 (1990)
17. del Cerro, L.F., Herzig, A.: A Modal Analysis of Possibility Theory. In: Jorrand, P., Kelemen, J. (eds.) FAIR 1991. LNCS, vol. 535, pp. 11–18. Springer, Heidelberg (1991)
18. Flaminio, T.: Strong non-standard completeness for fuzzy logic. Soft Comput. 12(4), 321–333 (2008)
19. Gabbay, D.: Theoretical foundations for non-monotonic reasoning in expert systems. In: Logics and Models of Concurrent Systems, pp. 439–457. Springer, Heidelberg (1985)
20. Grabisch, M., Labreuche, C.: Bi-capacities for decision making on bipolar scales. In: EUROFUSE Workshop in Information Systems, Varena, Italy, September 2002, pp. 185–190 (2002)
21. Godo, L., Marchioni, E.: Coherent conditional probability in a fuzzy logic setting. Logic Journal of the IGPL 14(3), 457–481 (2006)
22. Hailperin, T.: Sentential Probability Logic. Associated University Presses, Inc. (1996)
23. Hájek, P., Esteva, F., Godo, L.: Fuzzy logic and probability. In: Besnard, P., Hanks, S. (eds.) Proc. 11th Conf. Uncertainty in AI, Montreal, Canada, pp. 237–244 (1995)
24. Hájek, P.: Methamathematics of Fuzzy Logic. Kluwer Academic Publishers (1998)
25. Halpern, J., Pucella, R.: A logic for reasoning about evidence. Journal of Artificial Intelligence Research 26, 1–34 (2006)
26. Halpern, J.: Reasoning about Uncertainty. The MIT Press (2003)
27. Heifetz, A., Mongin, P.: Probability logic for type spaces. Games and Economic Behavior 35, 31–53 (2001)
28. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
29. Kunen, K.: Set theory - an introduction to independence proofs. Elsevier (1980)
30. Lehmann, D.: Generalized qualitative probability: Savage revisited. In: Horvitz, E., Jensen, F. (eds.) Procs. of 12 th Conference on Uncertainty in Artificial Intelligence (UAI 1996), pp. 381–388 (1996)
31. Lehmann, D., Magidor, M.: What does a conditional knowledge base entail? Artificial Intelligence 55, 1–60 (1992)
32. Lewis, D.: Counterfactuals. Basil Blackwell, London (1973)
33. Lukasiewicz, T.: Probabilistic Default Reasoning with Conditional Constraints. Annals of Mathematics and Artificial Intelligence 34, 35–88 (2002)
34. Lukasiewicz, T.: Weak nonmonotonic probabilistic logics. Artificial Intelligence 168(1-2), 119–161 (2005)
35. Lukasiewicz, T.: Nonmonotonic probabilistic logics under variable-strength inheritance with overriding: Complexity, algorithms, and implementation. International Journal of Approximate Reasoning 44(3), 301–321 (2007)
36. Marchioni, E., Godo, L.: A Logic for Reasoning about Coherent Conditional Probability: A Modal Fuzzy Logic Approach. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 213–225. Springer, Heidelberg (2004)

oarieli@mta.ac.il

37. Marchioni, E., Montagna, F.: On triangular norms and uninorms definable in Ł$\Pi\frac{1}{2}$. International Journal of Approximate Reasoning 47(2), 179–201 (2008)
38. Marker, D.: Model Theory: An Introduction. Springer, Heidelberg (2002)
39. Makinson, D.: General patterns in nonmonotonic reasoning. In: Handbook of Logic in Artificial Intelligence and Logic Programming. Non Monotonic Reasoning and Uncertain Reasoning, vol. 3, pp. 35–110. Clarendon Press, Oxford (1994)
40. Narens, L.: On qualitative axiomatizations for probability theory. Journal of Philosophical Logic 9(2), 143–151 (1980)
41. Nilsson, N.: Probabilistic logic. Artificial Intelligence 28, 71–87 (1986)
42. Ognjanović, Z., Rašković, M.: A logic with higher order probabilities. Publications de l'institut mathematique, Nouvelle série, tome 60(74), 1–4 (1996)
43. Ognjanović, Z., Rašković, M.: Some probability logics with new types of probability operators. J. Log. Comput. 9(2), 181–195 (1999)
44. Ognjanović, Z., Rašković, M.: Some first-order probability logics. Theoretical Computer Science 247(1-2), 191–212 (2000)
45. Ognjanović, Z., Marković, Z., Rašković, M.: Completeness Theorem for a Logic with imprecise and conditional probabilities. Publications de L'Institute Matematique (Beograd) 78(92), 35–49 (2005)
46. Ognjanović, Z.: Discrete linear-time probabilistic logics: completeness, decidability, complexity. J. Log. Comput. 16(2), 257–285 (2006)
47. Ognjanović, Z., Perović, A., Rašković, M.: Logic with the qualitative probability operator. Logic Journal of the IGPL 16(2), 105–120 (2008)
48. Ognjanović, Z., Doder, D., Marković, Z.: A Branching Time Logic with Two Types of Probability Operators. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 219–232. Springer, Heidelberg (2011)
49. Perović, A., Ognjanović, Z., Rašković, M., Marković, Z.: A Probabilistic Logic with Polynomial Weight Formulas. In: Hartmann, S., Kern-Isberner, G. (eds.) FoIKS 2008. LNCS, vol. 4932, pp. 239–252. Springer, Heidelberg (2008)
50. Perović, A., Ognjanović, Z., Rašković, M., Radojević, D.G.: Finitely additive probability measures on classical propositional formulas definable by Gödel's t-norm and product t-norm. Fuzzy Sets and Systems 169(1), 65–90 (2011)
51. Perović, A., Ognjanović, Z., Rašković, M., Marković, Z.: Qualitative Possibilities and Necessities. In: Sossai, C., Chemello, G. (eds.) ECSQARU 2009. LNCS (LNAI), vol. 5590, pp. 651–662. Springer, Heidelberg (2009)
52. Rašković, M.: Classical logic with some probability operators. Publications de l'institut mathematique, Nouvelle série, tome 53(67), 1–3 (1993)
53. Rašković, M., Ognjanović, Z.: A first order probability logic LP_Q. Publications de l'institut mathematique, Nouvelle série, tome 65(79), 1–7 (1999)
54. Rašković, M., Ognjanović, Z., Marković, Z.: A Logic with Conditional Probabilities. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 226–238. Springer, Heidelberg (2004)
55. van der Hoek, W.: Some considerations on the logic $P_F D$: a logic combining modality and probability. Journal of Applied Non-Classical Logics 7(3), 287–307 (1997)
56. Wellman, M.P.: Some varieties of qualitative probability. In: Proceedings of the 5th International Conference on Information Processing and the Management of Uncertainty, Paris (1994)
57. Zadeh, L.: Fuzzy sets. Information and Control 8(3), 338–353 (1965)
58. Zhu, Z., Xiao, W.: Two Representation Theorems for Non-monotonic Inference Relations. Journal of Logic and Computation 17(4), 727–747 (2007)
59. Zhu, Z., Zhang, D., Chen, S., Zhu, W.: Some contributions to nonmonotonic consequence. Journal of Computer Science and Technology 16(4), 297–314 (2001)

oarieli@mta.ac.il

Detecting Suspect Answers in the Presence of Inconsistent Information

Olivier Pivert[1] and Henri Prade[2]

[1] Irisa – Enssat, University of Rennes 1
Technopole Anticipa 22305 Lannion Cedex France
[2] IRIT, CNRS and University of Toulouse, 31062 Toulouse Cedex 9, France
pivert@enssat.fr, prade@irit.fr

Abstract. In the presence of inconsistent information, two classical approaches consist in (i) cleaning the information by means of an automated process, for instance by performing a minimal set of updates aimed at restoring consistency, (ii) returning only the answers that are certain with respect to a given query, as in consistent query answering. In this paper, we propose an alternative approach, somewhat inspired by artificial intelligence works, which is aimed at warning the user about the presence of suspect answers in a query result. Roughly speaking, the idea is that such elements can be identified inasmuch as they can be found in the answers to contradictory queries. This idea may also be refined by introducing some gradedness in terms of cardinality or similarity.

1 Introduction

Managing inconsistent information has become a very important issue at a time where the amount and the variety of pieces of information to be handled considerably increase the possibilities of inconsistency. For this reason, inconsistency, after uncertainty, has appeared on the agenda of both artificial intelligence and data bases research for more than a decade now. Still, this issue is addressed in a different perspective by the two research fields, even if in both cases, the proposed approaches can be roughly divided between those which try to restore or maintain consistency and those which accept inconsistent information trying to draw the "best" from it.

Thus, in artificial intelligence, inconsistency handling is a central issue in information fusion and in reasoning under inconsistency. In information fusion, e.g., [1], the requirement that the merging of individually consistent knowledge bases results into a consistent base is enforced when the bases to be fused are conflicting. When reasoning under inconsistency, restoring consistency is no longer compulsory, and different approaches have been proposed [2,3,4] for defining inconsistency-tolerant inferences which do not trivialize everything in case of inconsistent information (as classical logic does). In particular, the use of paraconsistent logics [5,6], which tolerate contradictory pieces of information, is of special interest in that respect. As we shall see, the inconsistency-tolerant approach presented in this paper can be viewed as reflecting an artificial intelligence spirit in the processing of queries addressed to an inconsistent database.

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 278–297, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

In the database world, there is a strong concern about the semantic consistency of stored data (wrt integrity constraints) and the avoidance of contradictory information. However, the support for specifying and enforcing semantic integrity in commercial products and practical systems remains scant [7].

Theoretically speaking, dirty databases should not exist, since declarative solutions — that have been widely proclaimed and advertised in the scientific and technical literature of the field — are available for specifying and enforcing integrity constraints [8]. However, in practice, dirty databases are quite common, for various reasons:

- the database may be ill-designed and/or some integrity constraints (in particular functional dependencies) are not properly enforced [7],
- when a database results from the integration of multiple data sources, or when the data comes from unverified sources or is uncertain, the resulting database generally contains inconsistencies [9].

Assessing the dirtiness of a database and devising methods to clean it or to overcome it are important problems which have been addressed from different points of view, see e.g. [10,11,12,13]. However, the measures of dirtiness that can be found in the literature are somewhat debatable from a semantic point of view [14]. Moreover, it is not quite clear how these measures can be used in practice and exploited in relation with cleaning methods. In this paper, we consider the case where dirtiness corresponds to the violation of one or several functional dependencies (FDs) — we do not deal with other forms of "dirtiness" such as null values, uncertain values, or the violation of other types of integrity constraints, even though this would of course make sense.

In the presence of inconsistent data, two points of view may be taken. The first one consists in cleaning the database so as to make it consistent, either by means of an automated process — see for instance [15,16,13,17,18] —, or by an interactive approach as that outlined in [14]. The second one — which is adopted in this paper and also in Consistent Query answering (CQA) approaches [19] — consists in taking into account the inconsistencies at query processing time.

The remainder of the paper is structured as follows. Section 2 introduces some notations used further and gives an example about FD violation. Section 3 discusses related works about database repair and data cleaning approaches. In Section 4, we take a pragmatic viewpoint and define a method — whose principle was briefly pointed out in [14] — for detecting suspect answers to projection-selection queries (which can be a valuable piece of information to provide the user with). Section 5 extends this approach to the case of semi-join queries. Section 6 is devoted to union queries. Section 7 concludes the paper and outlines perspectives for further research.

2 Notations and Introductory Example

We assume the existence of a relation r of schema $R = (A_1, \ldots, A_n)$ where the A_i's are attributes. Each attribute A_i has an associated domain, $dom(A_i)$.

oarieli@mta.ac.il

Table 1. An example database

	Name	*Age*	*Height*
t_1	Mary	28	170
t_2	Mary	28	176
t_3	John	30	163
t_4	John	30	160
t_5	John	30	164
t_6	Matthew	32	170
t_7	Paul	37	172
t_8	Paul	37	171
t_9	Paul	37	176
t_{10}	James	45	177

A tuple of r is a member of $dom(A_1) \times \ldots \times dom(A_n)$. The regular functional dependency $X \rightarrow Y$ is defined as:

$$\forall t_1,\ t_2 \in r,\ t_1.X = t_2.X \Rightarrow t_1.Y = t_2.Y \tag{1}$$

where X and Y denote sets of attributes. For instance, in Table 1, $Name \rightarrow Age$ is an FD saying that two tuples about the same person should agree on age. Such a table may result from the fusion of different potentially inconsistent data sources, which explains why the primary key constraint has been disabled. Notice that the FD $Name \rightarrow Height$ is violated by several pairs of tuples (for instance $(t_1,\ t_2)$).

Remark 1. Since any FD $X \rightarrow \{Y_1 \ldots Y_q\}$ can be decomposed into a set of FDs $\{X \rightarrow Y_1,\ \ldots,\ X \rightarrow Y_q\}$, it will be assumed in the following with no loss of generality that the right-hand side of an FD is made of a single attribute.

3 Related Work

The literature about data cleaning is quite rich, and in the following we only consider works which use data dependencies. These works are mainly oriented towards an *automated repair process*, which fundamentally distinguishes them from the type of approach considered in the present paper. The difference with consistent query answering will also be highlighted.

3.1 Automated Data Cleaning Approaches

Data cleaning [20] deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data.

In [15], the author deals with the *constraint repair problem* which attempts to bring a database in accordance with a given set of integrity constraints by applying modifications that are as small as possible. Unlike previous works on this topic, the approach proposed in [15] allows tuple *updates* as a repair primitive

oarieli@mta.ac.il

and shows that for conjunctive queries and a certain type of dependencies (called *full dependencies*), there exists a condensed representation of all repairs that permits computing trustable query answers.

In [16], Bohannon *et al.* define a database repair as a set of *value modifications* and introduce a cost framework that allows for the application of techniques from record-linkage to the search for good repairs. After proving that finding minimal-cost repairs in their model is NP-complete in the size of the database, the authors introduce an approach to heuristic repair-construction based on equivalence classes of attribute values and define two greedy algorithms.

In [13,17], the authors consider extensions of functional dependencies and inclusion dependencies, referred to as conditional functional dependencies (CFDs) and conditional inclusion dependencies (CINDs), respectively, by additionally specifying patterns of semantically related values. These patterns impose conditions on what part of the relation(s) the dependencies are to hold and which combinations of values should occur together. An example of CFD is $customer([cc = 44;\ zip] \rightarrow [street])$, which asserts that for customers in the UK ($cc = 44$), zip code determines street. It is an "FD" that is to hold on the subset of tuples that satisfies the pattern $cc = 44$. These constraints are then extended with similarity. In [21], the same authors present a research prototype system for data repairing which supports (a) specifications of CFDs, (b) automatic detections of CFD violations, relying on SQL-based techniques (which is an aspect this approach has in common with the method we describe further), and (c) repairing, i.e., given a set of CFDs and a dirty database, it finds a candidate repair that minimally differs from the original data and satisfies the CFDs [22].

In [18], the authors propose another automated data cleaning approach that deals with FD and CFD violations. They present an approximate method which, for a given set of FDs and a given inconsistent database, produces a repair whose distance to the initial database is in a constant factor with respect to the distance to the optimal repair.

In our opinion, the main drawback of these automated cleaning approaches lies in the fact that they may lead to a loss of potentially useful data. Since there are several ways of restoring consistency even in simple cases, it cannot be an objective *per se*. As a matter of fact, restoring consistency by means of a fully automated process implies taking the chance of introducing false data.

3.2 Consistent Query Answering

The starting point of consistent query answering (CQA for short) [10,23,11,24] is to consider that in many cases cleaning the database from inconsistencies may not be an option, e.g. in virtual data integration, or doing it may be costly, non-deterministic, and — as noted above — may lead to a loss of potentially useful data [23]. In consequence, CQA constitutes an alternative approach to data cleaning which consists in basically living with the inconsistent data, but making sure that the consistent data can be identified when queries are answered. In this approach, the first problem that has to be confronted is the one of characterizing in precise terms the notion of consistent data in a possibly inconsistent database.

oarieli@mta.ac.il

Such a definition was proposed in [10]. The basic intuition is that a piece of data in the database D is consistent if it is invariant under minimal forms of restoring the consistency of the database, i.e. it remains in every database instance D' that shares the schema with D, is consistent wrt the given ICs, and "minimally differs" from D [10]. This is similar to a basic idea in artificial intelligence that amounts, given an inconsistent knowledge base, to accept as valid conclusions all them that can be classically deduced from all its maximal consistent subbases [25]. Moreover, the idea of minimal change is also a key concept in knowledge revision and information fusion in artificial intelligence. Note also the similarity — which was pointed out e.g. in [26] — of the notion of consistent query answers to that of sure or certain answers in the context of incomplete databases [27].

4 Detecting Suspect Answers to Selection Queries

Hereafter, rather than identifying the certain answers to a given query, we aim at warning the user about the presence of suspect elements in the query result. Roughly speaking, the idea is that such elements can be identified inasmuch as they can be found in the answers to contradictory queries. This is similar to the idea in artificial intelligence to associate with a conclusion the information that it is paraconsistent, i.e., that the opposite conclusion could be supported as well, e.g. [4]. Moreover, note that a set of answers that does not contain any suspicious ones may still be an inconsistent set of information since FDs with respect to attributes not involved in the query may be violated in this set. Beside the fact that the approach we propose differs from CQA in the sense that it provides more information — which can be valuable, e.g. in the case where there are no (or very few) certain answers —, it is also much less costly since it avoids the combinatorics attached to the search for minimal repairs.

4.1 Culprits and Clusters

In [12], Martinez et al. introduce the notions of *culprits* and *clusters*. Culprits are the duals of the maximal consistent subsets (studied, e.g., in [28]). The following three definitions are drawn from [12].

Definition 1. Let r be a relation and F a set of functional dependencies. A *culprit* is a set $c \subseteq r$ such that $c \cup F$ is inconsistent and $\forall c' \subset c$, $c' \cup F$ is consistent.

Thus, culprits are minimal sets of database tuples that cause a functional dependency violation. In the case where a single FD $X \rightarrow Y$ is considered, every culprit is a pair of tuples that agree on X and disagree on Y. Let $culprits(r, fd)$ denote the set of culprits in relation r w.r.t. the functional dependency fd.

Example 1. Consider a functional dependency *fd* stating that

$$\forall t,\, t' \in r,\, t.Name = t'.Name \Rightarrow t.Height = t'.Height.$$

The relation in Table 1 has six culprits, namely $c_1 = \{t_1,\, t_2\}$, $c_2 = \{t_3,\, t_4\}$, $c_3 = \{t_3,\, t_5\}$, $c_4 = \{t_4,\, t_5\}$, $c_5 = \{t_7,\, t_8\}$, $c_6 = \{t_7,\, t_9\}$, and $c_7 = \{t_8,\, t_9\}$. $\diamond$

oarieli@mta.ac.il

Definition 2. Given two culprits c, $c' \in culprits(r,\ fd)$, we say that c and c' overlap, denoted by $c \,\Delta\, c'$, iff $c \cap c' \neq \emptyset$.

Definition 3. Let Δ^* be the reflexive transitive closure of relation Δ. A *cluster* is a set $cl = \bigcup_{c \in e} c$ where e is an equivalence class of Δ^*.

Example 2. In Table 1, the clusters are the sets $cl_1 = \{t_1,\ t_2\}$, $cl_2 = \{t_3,\ t_4,\ t_5\}$, and $cl_3 = \{t_7,\ t_8,\ t_9\}$. ◇

The notion of cluster will be used in the following, but it is important to emphasize that the approach proposed in [12] only aims at assessing the *degree of dirtiness* of a given database, whereas the objective we pursue is to identify the *suspect answers* to a query in the presence of inconsistent data.

4.2 Principle of the Approach

Let us consider a relation r of schema U. Let us consider a subset of U denoted by X (assumed to be a key of r), $\{Y_1, \ldots, Y_n\}$ the remaining attributes of U, and a query $Q = \pi_X(\sigma_\psi(r))$ which computes the projection over X of the result of the selection based on condition ψ. Let us assume that ψ involves conjunctions or disjunctions of elementary conditions of the form $Y_i \theta_i v_i$ where θ_i is a comparator and v_i is a constant. Let us denote by $res(Q)$ the result of Q, i.e., the set of tuples obtained when Q is run over the inconsistent database considered. Let us denote by $\mathcal{F} = \{X \rightarrow Y_1, \ldots, X \rightarrow Y_n\}$ the set of elementary FDs whose left-hand side is X. We consider the case where some of these FDs may be violated in r. In order to define the notion of a suspect answer, let us first introduce the concept of a *vp-repair*, i.e. a repair obtained through attribute *value propagation* inside every individual cluster. Let us emphasize that even though the well-known notion of repair is used for giving a clear semantics to the concept of a suspect answer in our approach, the technique we propose does *not* perform any data cleaning operation (nor does it *compute* any repair). This notion of a *potential* repair may be viewed as similar to the idea of *extension* which often appears in the semantics of non-classical artificial intelligence logics.

Definition 4. We call *vp-repair* of r a relation consistent in the sense of $\mathcal{F}$, obtained by cleaning every cluster of r. This cleaning is performed by updating the attribute values involved in the right-hand side of a violated FD in (some of) the tuples inside the cluster, using the attribute values present in the tuples of the same cluster. Let us denote by $A_{i,1}, \ldots, A_{i,m}$ the elementary attributes of U involved in an FD violation in a given cluster cl_i and by $cl_i[A_{i,j}]$ the projection of cluster cl_i over attribute $A_{i,j}$. Cleaning cl_i is done in the following way:

1. choose a given tuple $(v_1, \ldots, v_m)$ in $cl_i[A_{i,1}] \times \ldots \times cl_i[A_{i,m}]$,
2. for every tuple t_k of cl_i, for every $j \in \{1, \ldots, m\}$ do $t_k.A_{i,j} \leftarrow v_j$.

Example 3. Let us consider the relation r of schema $(X,\ Y_1,\ Y_2,\ Y_3)$, the set of FDs $\mathcal{F} = \{X \rightarrow Y_1,\ X \rightarrow Y_2,\ X \rightarrow Y_3\}$, and the following extension of r:

$$r = \{\langle x_1,\ 5,\ 130,\ 5000\rangle,\ \langle x_1,\ 10,\ 80,\ 5000\rangle\}.$$

oarieli@mta.ac.il

Here, the two tuples are grouped into a single cluster. There exist four *vp*-repairs of relation r:

$$r'_1 = \{\langle x_1,\ 5,\ 130,\ 5000\rangle,\ \langle x_1,\ 5,\ 130,\ 5000\rangle\}$$
$$r'_2 = \{\langle x_1,\ 10,\ 80,\ 5000\rangle,\ \langle x_1,\ 10,\ 80,\ 5000\rangle\}$$
$$r'_3 = \{\langle x_1,\ 5,\ 80,\ 5000\rangle,\ \langle x_1,\ 5,\ 80,\ 5000\rangle\}$$
$$r'_4 = \{\langle x_1,\ 10,\ 130,\ 5000\rangle,\ \langle x_1,\ 10,\ 130,\ 5000\rangle\}.$$ ◇

Remark 2. As illustrated by Example 3, the definition of a *vp*-repair does not consider that a tuple is "good" or "bad" as a whole, but rather that it may include correct values as well as wrong ones. This cautious view corresponds to acknowledging that a given source may be very reliable as to a certain attribute Y_1 and another source very competent when it comes to another attribute Y_2. The notion of a *vp*-repair is somewhat similar to that of an *uprepair* defined in [29] but is more restrictive since it only permits to modify the values of the attributes involved in the right-hand side of an FD (as do *LS*-repairs defined in [30] for key dependencies; let us recall that *LS*-repairs aim at minimizing the quantitative global distance between the original and modified instances of numerical attributes). It is also similar to the type of repairs considered in [16] except that these repairs may involve insertions whereas *vp*-repairs do not.

Definition 5. An answer x from $res(Q)$ is said to be consistent (or certain) iff for every *vp*-repair r' of r, x belongs to the result of $Q' = \pi_X(\sigma_\psi(r'))$. Let $cert(Q)$ be the set of the consistent answers to Q.

Remark 3. Note that a value $x = t.X$ such that t is a "suspicious tuple" w.r.t. the FD $X \rightarrow Y$ — i.e., $\exists t' \in r$ such that $t.X = t'.X \ \wedge\ t.Y \neq t'.Y$ — is not necessarily a suspicious answer to Q. A counter-example is $r = \{\langle John,\ 175\rangle,\ \langle John,\ 190\rangle\}$ where both tuples are suspicious w.r.t. the FD $Name \rightarrow Height$ but *John* is *not* a suspicious answer to the query: $\pi_{Name}(\sigma_{Height \geq 170}(r))$.

Definition 6. An answer x from $res(Q)$ is said to be *suspect* iff there exists a *vp*-repair r' of r such that x does not belong to the result of $Q' = \pi_X(\sigma_\psi(r'))$. Let $susp(Q)$ be the set of suspect answers to Q.

Remark 4. Obviously, $\langle cert(Q),\ susp(Q)\rangle$ forms a partition of $res(Q)$.

The following proposition establishes a link between the notion of a suspect answer and the negation of the original selection condition.

Proposition 1. *In case of a selection condition which is a conjunction of atomic predicates* $\psi = (Y_1\ \theta_1\ v_1) \wedge \ldots \wedge (Y_q\ \theta_q\ v_q)$*, one has:*

$$susp(Q) = (res(Q) \cap \pi_X(\sigma_{\neg\psi}(r))).$$

Proof. Let us first show that if $x \in (res(Q) \cap \pi_X(\sigma_{\neg\psi}(r)))$, then there exists a *vp*-repair r' of r such that x does not belong to the result of $Q' = \pi_X(\sigma_\psi(r'))$. In order to build such a *vp*-repair, it is sufficient to choose in $\sigma_{\neg\psi}(r')$ one of the tuples which have the value x and to use its Y_i-values for cleaning the cluster to

oarieli@mta.ac.il

which it belongs (as to the other clusters, the choice for cleaning is indifferent). Let us now consider a value x from $res(Q)$ such that there exists a *vp*-repair r' of r such that x does not belong to the result of $Q' = \pi_X(\sigma_\psi(r'))$. This means that there exists $i \in \{1, \ldots, q\}$ such that $X \rightarrow Y_i$ is supposed to hold and x is not in the result of $\pi_X(\sigma_{(Y_i\ \theta_i\ v_i)}(r'))$. Hence, this value x is necessarily in the result of $\pi_X(\sigma_{\neg(Y_i\ \theta_i\ v_i)}(r))$ and also in the result of $\pi_X(\sigma_{\neg\psi}(r))$. •

It is important to emphasize that the queries considered in Proposition 1 do not correspond to the whole class usually known as conjunctive queries, i.e., the fragment of first-order logic given by the set of formulas that can be constructed from atomic formulas using conjunction and existential quantification, whose general form is:

$$(x_1, \ldots, x_k).\exists x_{k+1}, \ldots x_m.A_1 \wedge \ldots \wedge A_h$$

where $A_1 \wedge \ldots \wedge A_h$ are atomic formulas. Let us recall that computing consistent answers to conjunctive queries is known to be co-NP complete in data complexity when there is even only one functional dependency and only deletions are allowed, and the same holds if only updates are allowed (as is the case considered in our approach).

On the other hand, the queries considered in Proposition 1, for which the method we propose is able to compute the certain answers in polynomial time (i.e., projection-selection queries involving a purely conjunctive selection condition) are more contrained since their general form is:

$$(x_1, \ldots, x_k).\exists x_{k+1}, \ldots x_m.r(x_1, \ldots, x_m) \wedge \psi_1 \wedge \ldots \wedge \psi_h$$

where r denotes a relation and each ψ_i is an atomic condition.

The following proposition, which generalizes Proposition 1, constitutes the basis of the detection algorithm.

Proposition 2. *For a query $Q = \pi_X(\sigma_\psi(r))$ involving a conjunctive selection condition $\psi = \psi_1 \wedge \ldots \wedge \psi_m$, one has:*

$$susp(Q) = res(Q) \cap \bigcup_{i=1..m} susp(Q_i)$$

where $Q_i = \pi_X(\sigma_{\psi_i}(r))$.

The proof is very similar to that of Proposition 1.

The following proposition provides a result about disjunctive conditions that will be used in the detection algorithm presented further.

Proposition 3. *When condition ψ is purely disjunctive, i.e., of the form $\psi = (Y_1\ \theta_1\ v_1) \vee \ldots \vee (Y_q\ \theta_q\ v_q)$, one has*

$$susp(Q) = res(Q) \cap \bigcap_{i=1..q} res(\overline{Q}_i)$$

where $\overline{Q}_i = \pi_X(\sigma_{\neg(Y_i\ \theta_i\ v_i)}(r))$.

oarieli@mta.ac.il

Proof. Let $x \in res(Q)$ be an element of $susp(Q)$. Then, there exists a *vp*-repair r' of r such that x does not belong to the result of $Q' = \pi_X(\sigma_\psi(r'))$. This implies that x is associated in r' with a set of values $(y_1, \ldots, y_q)$ such that $(y_1\ \theta_1\ v_1) \vee \ldots \vee (y_q\ \theta_q\ v_q)$ is false, i.e., such that $\forall i \in \{1, \ldots, q\}$, $(y_i\ \theta_i\ v_i)$ is false. Hence, in the cluster associated with x in r, for each i in $\{1, \ldots, q\}$, there exists a value y_i associated with x such that $(y_i\ \theta_i\ v_i)$ is false. Then, $\forall i \in \{1, \ldots, q\}$, $x \in res(\overline{Q}_i)$. Consequently, $x \in res(Q) \cap \bigcap_{i=1..q} res(\overline{Q}_i)$. Let now x be an element of $susp(Q)$, which means that x belongs to $\bigcap_{i=1..q} res(\overline{Q}_i)$. This implies that for every i in $\{1, \ldots, q\}$, x is associated in r with a value y_1 such that $(y_i\ \theta_i\ v_i)$ is false. Consequently, there exists a *vp*-repair r' of r such that x does not belong to the result of $Q' = \pi_X(\sigma_\psi(r'))$. This *vp*-repair can be constructed by choosing, for every attribute Y_i in the cluster associated with x a value y_i such that $(y_i\ \theta_i\ v_i)$ is false. •

4.3 Algorithmic Aspects

The general principle is as follows. One first writes the selection condition in conjunctive normal form. One then computes the suspect answers associated with each conjunct (which corresponds to a disjunction of atomic conditions) thanks to Proposition 3. Finally, the suspect answers to the global query are established thanks to Proposition 2.

Disjunction of atomic conditions. For a disjunctive condition of the form $\psi = (Y_1\ \theta_1\ v_1) \vee \ldots \vee (Y_q\ \theta_q\ v_q)$, the algorithm is as follows. Let us denote $Q = \pi_X(\sigma_\psi(r))$.

1. compute $res(Q)$;
2. $\forall i \in \{1, \ldots, q\}$, compute $res(\overline{Q_i})$ where $\overline{Q}_i = \pi_X(\sigma_{\neg(Y_i\ \theta_i\ v_i)}(r))$;
3. the set of certain answers to Q is $res(Q) - \bigcap_{i=1..q} res(\overline{Q}_i)$;
4. the set of suspect answers to Q is $res(Q) \cap \bigcap_{i=1..q} res(\overline{Q}_i)$.

The proof of its soundness and completeness is straightforwardly based on Proposition 3.

Conjunctive condition. Let us consider a query $Q = \pi_X(\sigma_\psi(r))$ with $\psi = \psi_1 \wedge \ldots \wedge \psi_m$. The global algorithm is as follows.

1. compute $res(Q)$;
2. for each i in $\{1, \ldots, m\}$ compute $S_i = susp(\pi_X(\sigma_{\psi_i}(r))$ using the procedure given above;
3. according to Proposition 2, the set of suspect answers to Q is

$$res(Q) \cap \bigcup_{i=1..m} S_i).$$

The proof of its soundness and completeness is based on Propositions 2 and 3.

oarieli@mta.ac.il

Table 2. Relation *Emp* (Examples 4 and 5)

	Name	*Age*	*Height*	*City*
t_1	Mary	32	170	Chicago
t_2	Mary	34	176	Boston
t_3	John	30	163	Chicago
t_4	John	32	160	Chicago
t_5	John	30	164	Chicago
t_6	Matthew	32	170	Dallas
t_7	Matthew	38	170	Austin
t_8	Paul	29	173	Chicago
t_9	Paul	37	170	Chicago
t_{10}	Paul	27	170	Boston
t_{11}	James	45	177	Chicago
t_{12}	James	43	187	Chicago

Example 4. Let us consider relation *Emp* from Table 2 along with the query

$$Q = \pi_{Name}(\sigma_{((Age>31 \vee Height>171) \wedge City='Chicago')}(Emp)).$$

The FDs $\{Name \rightarrow Age, Name \rightarrow Height, Name \rightarrow City\}$ are supposed to be valid on relation *Emp*. One gets:

- $res(Q) = \{$Mary, John, Paul, James$\}$;
- $res(\pi_X(\sigma_{(Age>31 \vee Height>171)}(Emp)) =$ $\{$Mary, John, Matthew, Paul, James$\}$;
- $res(\pi_X(\sigma_{(Age\leq 31)}(Emp)) = \{$John, Paul$\}$;
- $res(\pi_X(\sigma_{(Height\leq 171)}(Emp)) = \{$Mary, John, Matthew, Paul$\}$;
- Thus, $susp(\pi_X(\sigma_{(Age>31 \vee Height>171)}(Emp))) = \{$John, Paul$\}$;
- $res(\pi_X(\sigma_{(City='Chicago')}(Emp)) = \{$Mary, John, Paul, James$\}$;
- $res(\pi_X(\sigma_{(City\neq'Chicago')}(Emp)) = \{$Mary, Matthew, Paul$\}$;
- Thus, $susp(\pi_X(\sigma_{(City='Chicago')}(Emp))) = \{$Mary, Paul$\}$;
- Finally,

$$\begin{aligned} susp(Q) &= \{\text{Mary, John, Paul, James}\} \cap (\{\text{John, Paul}\} \cup \{\text{Mary, Paul}\}) \\ &= \{\text{Mary, John, Paul}\}. \end{aligned}$$

The only certain answer is $\langle$James$\rangle$.

Notice that with a CQA approach, only $\langle$James$\rangle$ would be returned since it is the only certain answer to Q (which illustrates the gain in terms of informativeness of the approach we propose). ◇

Remark 5. The detection process may be extended to queries of the form

$$Q = \pi_{XZ}(\sigma_{\psi}(r))$$

where Z denotes a set of attributes disjoint from X. Then, for every answer xz to Q, one must check whether x is suspect or not. If x is suspect, then all of the answers involving x are also suspect. The same applies to Z if there is an FD between Z and one of the Y_i's.

oarieli@mta.ac.il

Example 5. Let us consider again relation *Emp* from Example 4 and its content represented in Table 2, as well as the query:

$$Q = \pi_{\{Name,\ Age\}}(\sigma_{Height>172}(Emp)).$$

The answers to Q are ⟨Mary, 34⟩, ⟨Paul, 29⟩, ⟨James, 45⟩, and ⟨James, 43⟩. Using the query:

$$Q = \pi_{\{Name\}}(\sigma_{Height\leq 172}(Emp))$$

one detects that Mary and Paul are suspect, which implies that the only non-suspect answers to Q are ⟨James, 45⟩ and ⟨James, 43⟩. ◇

4.4 A Gradual View Based on Cardinality

In the case of a conjunction of atomic conditions, one may think of refining the process described above by indicating *how much* a tuple is suspect. The idea is to consider that a value xz is all the more suspect as the number of results $\pi_X(\sigma_{\neg\psi_i}(r))$ to which it belongs is high, where ψ_i denotes one of the conjuncts of ψ. However, this makes the computation significantly more costly. The algorithm is as follows:

1. process Q and store its result in a temporary table res;
2. for every conjunct ψ_i of ψ, run the query $\pi_X(\sigma_{\neg\psi_i}(r))$ and store its result in a temporary table neg_i;
3. display the answers which have not been detected as suspect by means of the query $sp_0 = res - (neg_1 \cup \ldots \cup neg_k)$, where k denotes the number of conjuncts in ψ;
4. find the very suspect answers, i.e., those returned by negating all of the conjuncts of ψ, using

$$sp_k = res \cap neg_1 \cap \ldots \cap neg_k,$$

then the answers a little bit less suspect (those obtained by negating $k-1$ conjuncts of ψ) using

$$\begin{aligned} sp_{k-1} = &((res \cap neg_1 \cap \ldots \cap neg_{k-1}) \cup \\ &(res \cap neg_1 \cap \ldots \cap neg_{k-2} \cup neg_k) \cup \\ &\ldots \\ &(res \cap neg_2 \cap \ldots \cap neg_k)) - sp_k \end{aligned}$$

etc, up to the least suspect answers (those retrieved by negating only one conjunct of ψ) using

$$sp_1 = ((res \cap neg_1) \cup (res \cap neg_2) \cup \ldots (res \cap neg_k)) - sp_2.$$

One first displays the content of the temporary table sp_1, then that of sp_2, etc, and finally that of sp_k, and indicates each time the risk level associated with a given group of answers.

oarieli@mta.ac.il

Table 3. Relation *Emp* (Example 6)

	Name	*Age*	*Height*	*City*
t_1	Mary	28	170	Boston
t_2	Mary	38	176	Chicago
t_3	John	30	173	Chicago
t_4	John	32	160	Chicago
t_5	John	33	163	Chicago
t_6	Matthew	35	170	Dallas
t_7	Matthew	32	170	Dallas
t_8	Paul	37	175	Chicago
t_9	Paul	27	185	Chicago
t_{10}	Paul	37	176	Chicago
t_{11}	James	45	177	Chicago

Obviously, the drawback of this method is that it is combinatorial, but in practice, one may think that k will be low in general. Anyway, the good thing is that this technique still has a polynomial *data complexity*.

Example 6. Let us consider relation *Emp* represented in Table 3 over which the functional dependencies $\mathcal{F} = \{Name \rightarrow Age, Name \rightarrow Height, Name \rightarrow City\}$ are supposed to be valid. Let us consider the query

$$\pi_{Name}(\sigma_{(Age>30 \,\wedge\, Height>171 \,\wedge\, City=\text{`}Chicago\text{'})}(Emp')).$$

Its answers are ordered as follows:

$\langle James \rangle$ nonsuspect answer,
$\langle Paul \rangle$ suspect answer with one FD violated,
$\langle John \rangle$ suspect answer with two FDs violated,
$\langle Mary \rangle$ suspect answer with three FDs violated. ◇

4.5 A Gradual View Based on the Intensity of the Violations

Another point of view for assessing the extent to which an answer x is suspect consists in using a proximity relation $\mu_{prox_{Y_i}}$ defined on the domain of each attribute Y_i present in $\mathcal{F}$. Let us use the notation:

$$\mu_i(x) = 1 - min_{(t,\, t') \in r \,|\, t.X = t'.X = x \,\wedge\, x \in (res \,\cap\, neg_i)}\, \mu_{prox_{Y_i}}(t.Y_i,\ t'.Y_i),$$

which corresponds to the maximal degree of "divergence" on Y_i (with respect to the FD $X \rightarrow Y_i$) among the tuples t such that $t.X = x$.

Let us build a vector $V(x) = \langle \alpha_1, \ldots, \alpha_n \rangle$ such that α_i is the ith highest degree among the $\mu_i(x)$'s. The answers may be ranked according to the lexicographic order applied to these vectors so as to first display the most suspect ones (but one also may choose to first display the least suspect ones by using rather $1 - \mu_i(x)$).

oarieli@mta.ac.il

Table 4. Relation *Emp* (Example 7)

	Name	*Age*	*Height*	*City*
t_1	Mary	28	170	Boston
t_2	Mary	38	176	Boston
t_3	John	30	173	Chicago
t_4	John	32	160	Chicago
t_5	John	33	163	Chicago
t_6	Matthew	35	170	Dallas
t_7	Matthew	32	170	Dallas
t_8	Paul	37	172	Seattle
t_9	Paul	27	171	Seattle
t_{10}	Paul	37	176	Seattle
t_{11}	James	45	177	Atlanta

Example 7. Let us consider relation Emp' represented in Table 4 over which the functional dependencies $\mathcal{F} = \{Name \rightarrow Age, Name \rightarrow Height, Name \rightarrow City\}$ are supposed to be valid. Let us define the proximity relations on the domains of *Age* and *Height* respectively as follows:

$$\mu_{prox_{Age}}(a,\ b) = \begin{cases} 0 \text{ if } |a-b| \geq 10, \\ 1 - \frac{|a-b|}{10} \text{ otherwise.} \end{cases}$$

$$\mu_{prox_{Height}}(a,\ b) = \begin{cases} 0 \text{ if } |a-b| \geq 20, \\ 1 - \frac{|a-b|}{20} \text{ otherwise.} \end{cases}$$

Let us consider the query

$$\pi_{Name}(\sigma_{(Age>30\ \wedge\ Height>171)}(Emp')).$$

Its answers are ordered as follows (from the most suspect ones to the least suspect ones):

$\langle Mary \rangle$ suspect answer with vector $\langle 1,\ 0.3 \rangle$,
$\langle Paul \rangle$ suspect answer with vector $\langle 1,\ 0.25 \rangle$,
$\langle John \rangle$ suspect answer with vector $\langle 0.65,\ 0.3 \rangle$,
$\langle James \rangle$ nonsuspect answer. ◇

5 Dealing with Semi-join Queries

Let us now consider the case where the query dealt with involves a semi-join (let us recall that this operation returns the tuples of the left-hand side relation which join with at least one tuple from the right-hand side relation). Let us consider the relations r and s of respective schemas $R(X,\ Y,\ Z)$ and $S(Y,\ W)$. Let us assume that r and s are equipped with the FDs $X \rightarrow Y$ and $Y \rightarrow W$ respectively (but these FDs may be violated). Let us consider the query

$$Q = \pi_X(\sigma_{\psi_1}(r) \ltimes_{Y=Y} \sigma_{\psi_2}(s))$$

oarieli@mta.ac.il

where ψ_1 (resp. ψ_2) is a conjunction of elementary conditions supposed to concern Y (resp. W). Query Q aims at retrieving the X-values present in r in tuples which (i) satisfy condition ψ_1 and (ii) contain a Y-value which is present in s in at least one tuple satisfying condition ψ_2.

Example 8. Let us consider the relations *Emp* and *Pop* of respective schemas (*Name*, *Age*, *City*) and (*City*, *NbInhab*). The FD $Name \rightarrow City$ (resp. $City \rightarrow NbInhab$) is supposed to be valid in *Emp* (resp. *Pop*). Let us consider the semi-join query looking for those employees who live in a French city of less than 100,000 inhabitants. It is expressed as:

$$\pi_{Name}(\sigma_{French(City)}(Emp) \ltimes_{City=City} \sigma_{NbInhab<100K}(Pop)). \qquad \diamond$$

Definition 7. An answer x to Q is suspect iff there exists a *vp*-repair r' of r and a *vp*-repair s' of s such that x does not belong to the result of

$$Q = \pi_X(\sigma_{\psi_1}(r') \ltimes_{Y=Y} \sigma_{\psi_2}(s')).$$

Let us now consider the detection process based on the following proposition.

Proposition 4. *An answer x to Q is suspect if it is also in the result of the query:*

$$Q' = \pi_X(\sigma_{\neg\psi_1}(r)) \cup \pi_X(r \ltimes_{Y=Y} \sigma_{\neg\psi_2}(s)) \cup \pi_X(r \ltimes_{Y=Y} (\pi_Y(r) - \pi_Y(s))).$$

In other words, the values x which are suspect in $res(Q)$ are those which

- either are associated in r whith a Y-value which does not satisfy condition ψ_1, or
- are associated in r with a Y-value which is associated in s with a W-value which does not satisfy condition ψ_2, or
- are associated in r with a Y-value which is not present in s.

Table 5. Relations *Emp* (left) and *Pop* (right)

	Name	*Age*	*City*
t_1	John	28	Montauban
t_2	John	28	Bath
t_3	Mary	25	Cannes
t_4	Mary	25	Lannion
t_5	Paul	30	Pau
t_6	Paul	30	Beauvais
t_7	Louis	32	Lannion
t_8	David	45	Guingamp
t_9	David	45	Lannion
t_{10}	Erwan	52	Rennes

	City	*NbInhab*
s_1	Montauban	53,300
s_2	Cannes	70,400
s_3	Cannes	170,400
s_4	Pau	82,500
s_5	Pau	92,500
s_6	Beauvais	54,100
s_7	Lannion	21,300
s_8	Bath	83,990
s_9	Rennes	206,650

oarieli@mta.ac.il

Example 9. Let us consider the query from Example 8 and the relations *Emp* and *Pop* represented in Table 5. The answers to Q which are suspect are:

- John (since he may live in a nonFrench city, namely Bath); it is retrieved by the first subquery of Q' (cf. Proposition 4), namely:

$$\pi_{Name}(\sigma_{\neg French(City)}(Emp));$$

- Mary (since Cannes may have a population over 100,000); it is retrieved by the second subquery of Q' (cf. Proposition 4), namely:

$$\pi_{Name}(Emp \ltimes_{City=City} \sigma_{NbInhab \geq 100K}(Pop));$$

- David (since he may live in a city which is not present in *Pop*); it is retrieved by the third subquery of Q' (cf. Proposition 4), namely:

$$\pi_{Name}(Emp \ltimes_{City=City} (\pi_{City}(Emp) - \pi_{City}(Pop))).$$

The nonsuspect answers to Q are Paul and Louis. ◇

Let us now prove the validity of Proposition 4.

Proof. Let us consider the three cases of suspect answers retrieved thanks to the process outlined in Proposition 4.

- $x \in \pi_X(\sigma_{\neg\psi_1}(r))$: let us consider the *vp*-repair r_1 of r built by cleaning the cluster containing x using (one of) the Y-value(s) associated with x in r which does (do) not satisfy ψ_1. Then, whatever the *vp*-repair s' of s, x is not in the result of $Q = \pi_X(\sigma_{\psi_1}(r_1) \ltimes_{Y=Y} \sigma_{\psi_2}(s'))$. For instance, considering the relations from Table 5, this is what happens for John when the cluster $\{t_1, t_2\}$ of *Emp* is replaced by ⟨John, 28, Bath⟩.
- $x \in \pi_X(r \ltimes_{Y=Y} \sigma_{\neg\psi_2}(s))$: there exists a value y associated with x in r such that y is associated in s with a W-value w which does not satisfy ψ_2. Let us consider the *vp*-repair s_1 of s built by cleaning the cluster containing such a y using such a w and the *vp*-repair r_2 of r built by cleaning the cluster containing x using the same value y. Then, x is not in the result of $Q = \pi_X(\sigma_{\psi_1}(r_2) \ltimes_{Y=Y} \sigma_{\psi_2}(s_1))$. For instance, considering the relations from Table 5, this is what happens for Mary when the cluster $\{t_3, t_4\}$ of *Emp* is replaced by ⟨Mary, 25, Cannes⟩ and the cluster $\{s_2, s_3\}$ of *Pop* is replaced by ⟨Cannes, 170,400⟩.
- $x \in \pi_X(r \ltimes_{Y=Y} (\pi_Y(r) - \pi_Y(s)))$: there exists a Y-value y which is associated with x in r but which is absent from s. Let us consider the *vp*-repair r_2 of r built by cleaning the cluster containing x using such a y. Then, whatever the *vp*-repair s' of s, x is not in the result of $Q = \pi_X(\sigma_{\psi_1}(r_2) \ltimes_{Y=Y} \sigma_{\psi_2}(s'))$. For instance, considering the relations from Table 5, this is what happens for David when the cluster $\{t_1, t_2\}$ of *Emp* is replaced by ⟨David, 45, Guingamp⟩. •

oarieli@mta.ac.il

The detection process can rely on an algorithm similar to that presented in Subsection 4.3. The difference is that the "negative" query involves three subqueries.

Remark 6. In the special case where there is no $Y \rightarrow W$ in s, the detection of the suspect answers to Q may be performed by means of the query:

$$Q' = \pi_X(\sigma_{\neg\psi_1}(r)) \cup \pi_X(r \ltimes_{Y=Y} (\pi_Y(r) - \pi_Y(s)))$$

which corresponds to the query from Proposition 4 minus its second subquery. Indeed, since $X \rightarrow W$ does not have to be enforced anymore, an answer to x may perfectly be associated in r with a Y-value which is in a tuple from s that does not satisfy condition ψ_2 (as long as this Y-value is in another tuple from s satisfying ψ_2).

6 Union Queries

We now deal with a special case of union queries for which suspect answers can be detected. Namely, let us consider union queries of the form $Q = Q_1 \cup Q_2$ where $Q_1 = \pi_{X_1}(\sigma_{\psi_1}(r))$ and $Q_2 = \pi_{X_2}(\sigma_{\psi_2}(r))$ where ψ_1 and ψ_2 are conjunctions of atomic conditions of the form *attribute* θ *value* (θ being a comparator). As usual, it is assumed that the results of Q_1 and Q_2 (denoted respectively by res_1 and res_2) are relations that are union-compatible, i.e. X_1 and X_2 have the same number of attributes and identical domains for corresponding attributes. For the sake of simplicity, we will assume that the attributes are the same in X_1 and X_2 and we will use the notation X. The schema of r is of the form $\{X, Y_{r_1}, \ldots, Y_{r_m}\}$ and that of s of the form $\{X, Y_{s_1}, \ldots, Y_{s_{m'}}\}$. Condition ψ_1 (resp. ψ_2) concerns a subset Y_1 (resp. Y_2) of $\{Y_{r_1}, \ldots, Y_{r_m}\}$ (resp. $\{Y_{s_1}, \ldots, Y_{s_{m'}}\}$. Let us suppose that r (resp. s) is equipped with a set of FDs $\mathcal{F}_r = \{X \rightarrow Y_{r_1}, \ldots, X \rightarrow Y_{r_m}\}$ (resp. $\mathcal{F}_s = \{X \rightarrow Y_{s_1}, \ldots, X \rightarrow Y_{s_{m'}}\}$) — which may be violated. Let us denote $Y = \{Y_{r_1}, \ldots, Y_{r_m}\} \cap \{Y_{s_1}, \ldots, Y_{s_{m'}}\}$. If $Y \neq \emptyset$, suspect answers to Q may be detected. Even if there is no FD violation in r and s, there may be some in $res_1 \cup res_2$. Let us denote by $\overline{Q_1}$ and $\overline{Q_2}$ the queries that may be used for detecting the suspect answers in res_1 and res_2 respectively (cf. Section 4.2).

Proposition 5. *An answer x to Q is suspect if it is suspect in res_1 (which can be checked by means of $\overline{Q_1}$) or it is suspect in res_2 (which can be checked by means of $\overline{Q_2}$) or it belongs to the result of the query:*

$$\overline{Q_3} = (\pi_X(\sigma_{\psi'_1 \wedge \neg\psi'_2}(r)) \cap \pi_X(\sigma_{\psi'_2}(s))) \cup (\pi_X(\sigma_{\psi'_1}(r)) \cap \pi_X(\sigma_{\psi'_2 \wedge \neg\psi'_1}(s)))$$

where ψ'_1 (resp. ψ'_2) denotes the subconjunction of ψ_1 (resp. ψ_2) that involves the conditions of ψ_1 (resp. ψ_2) which concern attributes from Y_1 (resp. Y_2).

Here, the detection process is a bit more tricky than in the previous cases since one has to detect the suspect answers to subqueries first, which implies annotating the suspect tuples in res_1 and res_2.

oarieli@mta.ac.il

Proof. Let us consider a answer x to Q which belongs to the result of $\overline{Q_3}$. If x belongs to $\pi_X(\sigma_{\psi'_1 \wedge \neg\psi'_2}(r)) \cap \pi_X(\sigma_{\psi'_2}(s))$, then it means that x is present in r and s with two different Y-values, one in s which satisfies ψ'_2, one in r which does not. Tuple x is thus a suspect answer to Q. If x belongs to $\pi_X(\sigma_{\psi'_1}(r)) \cap \pi_X(\sigma_{\psi'_2 \wedge \neg\psi'_1}(s)))$, the principle is the same and one may deduce that x is associated in r and s with two different Y-values, which implies again that x is a suspect answer to Q. •

Table 6. Relations E_1 (left) and E_2 (right)

	Name	*Age*	*Height*	*Dept*
t_1	Mary	33	170	Sales
t_2	John	28	163	Sales
t_3	John	36	163	Sales
t_4	John	30	163	Sales
t_5	Matthew	32	170	Research
t_6	Matthew	31	170	Sales
t_7	Paul	37	170	Sales
t_8	James	45	177	Sales
t_9	Peter	51	185	Marketing
t_{10}	Richard	33	180	Sales

	Name	*Age*	*City*
s_1	Mary	37	Boston
s_2	Mary	41	Boston
s_3	John	36	Boston
s_4	John	36	Dallas
s_5	Matthew	30	Boston
s_6	Paul	37	Boston
s_7	Paul	37	Chicago
s_8	Paul	37	Boston
s_9	Greg	45	Boston

Example 10. Let us consider the query:

$$\pi_{Name}(\sigma_{Age>30 \wedge Dept='Sales'}(E_1)) \cup \pi_{Name}(\sigma_{Age>35 \wedge City='Boston'}(E_2))$$

and the relations E_1 and E_2 represented in Table 6. Relations E_1 and E_2 are assumed to be equipped with the respective sets of FDs $\mathcal{F}_1 = \{Name \rightarrow Age, Name \rightarrow Height, Name \rightarrow Dept\}$ and $\mathcal{F}_2 = \{Name \rightarrow Age, Name \rightarrow City\}$. Let us denote by $susp_1$ (resp. $susp_2$) the set of suspect tuples in res_1 (resp. res_2). These sets can be determined using the following queries:

- $\overline{Q_1} = \pi_{Name}(\sigma_{(Age\leq 30 \vee Dept\neq 'Sales')}(E_1)$,
- $\overline{Q_2} = \pi_{Name}(\sigma_{(Age\leq 35 \vee City\neq 'Boston')}(E_2)$.

For each answer x to Q which is not in $susp_1 \cup susp_2$, one has to check whether x belongs to the result of the following query:

$$\overline{Q_3} = (\pi_{\{Name\}}(\sigma_{Age>30 \wedge Age\leq 35}(E_1)) \cap \pi_{\{Name\}}(\sigma_{Age>35}(E_2))) \cup (\pi_{\{Name\}}(\sigma_{Age>30}(E_1)) \cap \pi_{\{Name\}}(\sigma_{Age>35 \wedge Age\leq 30}(E_2)))$$

which can be simplified into:

$$\overline{Q_3} = (\pi_{\{Name\}}(\sigma_{Age>30 \wedge Age\leq 35}(E_1)) \cap \pi_{\{Name\}}(\sigma_{Age>35}(E_2))).$$

Mary, for instance, is an answer to Q which belongs neither to the result of $\overline{Q_1}$ nor to that of $\overline{Q_2}$, but it belongs to the result of $\overline{Q_3}$ and is thus a suspect answer. Finally, the suspect answers to Q are {Mary, John, Matthew, Paul} and the nonsuspect ones are {James, Richard, Greg}. ◇

oarieli@mta.ac.il

7 Conclusion

In this paper, we have dealt with the problem raised by the presence of FD violations in a database. The main contribution lies in the proposal of an approach *which does not aim at automatically cleaning a database* (which could lead to a loss of useful information), but rather at warning the user in case a query result contains suspicious answers (i.e., elements related to inconsistencies). Roughly speaking, the idea is that such elements can be identified inasmuch as they can be found in the answers to contradictory queries. This idea can also be refined by taking into account graded aspects related to cardinality and similarity.

Several perspectives can be thought of, among which:

- extension of the approach to other kinds of queries;
- generalization to other kinds of integrity constraints than functional dependencies, in particular association constraints, whose basic form is:

$$\forall t \in r,\ t.A_1\ \theta_1\ v_1\ \wedge\ \ldots\ \wedge\ t.A_m\ \theta_m\ v_m \Rightarrow t.A_p\ \theta_p\ v_p$$

  where A_i is an attribute, θ_i a comparator, and v_i a constant. An example is "a person with age below 2 years should have a weight below 50 pounds".
- investigation of the link between the AI work about logics of inconsistency handling and a logical view of the proposed approach.

In the approach presented in this paper, the suspect nature of some answers to a request is identified through auxiliary queries. This may be viewed as a form of cooperative answers where additional information (here, the suspect nature of an answer, possibly with a degree) is provided to the user. Similar ideas may be found in other works as well. Thus in [31], the authors take advantage of the lineage of answers for finding causes for a query and computing a degree of responsibility of a tuple with respect to an answer, as a basis for explaining unexpected answers to a query. The idea there is that "Tuples with high responsibility tend to be interesting explanations to query answers". Another example of explanation needs is when the set of answers obtained can be clustered in clearly distinct subsets of similar or close answers. Then, it may be interested for the user to know what meaningful differences exist between the tuples leading to the answers that may explain the discrepancy in the result. For instance, if we look for possible prices for houses to let obeying some specifications, and that two clusters of prices are found, one may discover, e.g., that this is due to two categories of houses having, or not, some additional valuable equipments; see [32]. This general issue of providing answers with additional information about confidence, relevance, or explanations is certainly a challenging research direction.

References

1. Konieczny, S., Grégoire, E.: Logic-based information fusion in artificial intelligence. Information Fusion 7(1), 4–18 (2006)
2. Gabbay, D., Hunter, T.: Making Inconsistency Respectable (part 1). In: Jorrand, P., Kelemen, J. (eds.) FAIR 1991. LNCS (LNAI), vol. 535, pp. 19–32. Springer, Heidelberg (1991)

oarieli@mta.ac.il

3. Besnard, P., Hunter, A.: Reasoning with Actual and Potential Contradictions. In: Gabbay, D.M., Smets, P. (eds.) Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol. 2. Springer, Heidelberg (1998)
4. Benferhat, S., Dubois, D., Prade, H.: An overview of inconsistency-tolerant inferences in prioritized knowledge bases. In: Dubois, D., Prade, H., Klement, E. (eds.) Fuzzy Sets, Logic and Reasoning about Knowledge. Applied Logic Series, vol. 15, pp. 395–417. Kluwer, Dordrecht (1999)
5. Carnielli, W.A., Coniglio, M.E., Marcos, J.: Logics of formal inconsistency. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn., vol. 14, pp. 1–93. Springer, Heidelberg (2007)
6. Besnard, P., Hunter, A.: Quasi-Classical Logic: Non-Trivializable Classical Reasoning from Inconsistent Information. In: Froidevaux, C., Kohlas, J. (eds.) ECSQARU 1995. LNCS, vol. 946, pp. 44–51. Springer, Heidelberg (1995)
7. Decker, H., Martinenghi, D.: Getting rid of straitjackets for flexible integrity checking. In: DEXA Workshops, pp. 360–364. IEEE Computer Society (2007)
8. Martinenghi, D., Christiansen, H., Decker, H.: Integrity checking and maintenance in relational and deductive databases and beyond. In: Ma, Z. (ed.) Intelligent Databases: Technologies and Applications, pp. 238–285. Idea Group (2006)
9. Decker, H., Martinenghi, D.: Avenues to flexible data integrity checking. In: DEXA Workshops, pp. 425–429. IEEE Computer Society (2006)
10. Arenas, M., Bertossi, L.E., Chomicki, J.: Answer sets for consistent query answering in inconsistent databases. TPLP 3(4-5), 393–424 (2003)
11. Wijsen, J.: Project-Join-Repair: An Approach to Consistent Query Answering Under Functional Dependencies. In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) FQAS 2006. LNCS (LNAI), vol. 4027, pp. 1–12. Springer, Heidelberg (2006)
12. Martinez, M.V., Pugliese, A., Simari, G.I., Subrahmanian, V.S., Prade, H.: How Dirty Is Your Relational Database? An Axiomatic Approach. In: Mellouli, K. (ed.) ECSQARU 2007. LNCS (LNAI), vol. 4724, pp. 103–114. Springer, Heidelberg (2007)
13. Bohannon, P., Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for data cleaning. In: Proc. of ICDE, pp. 746–755 (2007)
14. Pivert, O., Prade, H.: Handling Dirty Databases: From User Warning to Data Cleaning — Towards an Interactive Approach. In: Deshpande, A., Hunter, A. (eds.) SUM 2010. LNCS (LNAI), vol. 6379, pp. 292–305. Springer, Heidelberg (2010)
15. Wijsen, J.: Condensed Representation of Database Repairs for Consistent Query Answering. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 375–390. Springer, Heidelberg (2002)
16. Bohannon, P., Flaster, M., Fan, W., Rastogi, R.: A cost-based model and effective heuristic for repairing constraints by value modification. In: Proc. of SIGMOD, pp. 143–154 (2005)
17. Fan, W., Geerts, F., Jia, X.: Conditional Dependencies: A Principled Approach to Improving Data Quality. In: Sexton, A.P. (ed.) BNCOD 2009. LNCS, vol. 5588, pp. 8–20. Springer, Heidelberg (2009)
18. Kolahi, S., Lakshmanan, L.V.S.: On approximating optimum repairs for functional dependency violations. In: Proc. of ICDT 2009, pp. 53–62 (2009)
19. Bertossi, L.: Database Repairing and Consistent Query Answering. Synthesis Lecture Notes on Data Management. Morgan & Claypool (2011)
20. Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. IEEE Data Eng. Bull. 23(4), 3–13 (2000)

oarieli@mta.ac.il

21. Fan, W., Geerts, F., Jia, X.: Semandaq: a data quality system based on conditional functional dependencies. PVLDB 1(2), 1460–1463 (2008)
22. Cong, G., Fan, W., Geerts, F., Jia, X., Ma, S.: Improving data quality: Consistency and accuracy. In: Proc. VLDB, pp. 315–326 (2007)
23. Bertossi, L.E.: Consistent query answering in databases. SIGMOD Record 35(2), 68–76 (2006)
24. Chomicki, J.: Consistent Query Answering: Five Easy Pieces. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 1–17. Springer, Heidelberg (2006)
25. Rescher, N., Manor, R.: On inference from inconsistent premises. Theory and Decision 1, 179–219 (1970)
26. Chomicki, J.: Consistent Query Answering: The First Ten Years. In: Greco, S., Lukasiewicz, T. (eds.) SUM 2008. LNCS (LNAI), vol. 5291, pp. 1–3. Springer, Heidelberg (2008)
27. Lipski, W.: On semantic issues connected with incomplete information databases. ACM Transactions on Database Systems 4(3), 262–296 (1979)
28. Baral, C., Kraus, S., Minker, J., Subrahmanian, V.S.: Combining knowledge bases consisting of first-order analysis. Computational Intelligence 8, 45–71 (1992)
29. Wijsen, J.: Database repairing using updates. ACM Trans. Database Syst. 30(3), 722–768 (2005)
30. Bertossi, L.E., Bravo, L., Franconi, E., Lopatenko, A.: The complexity and approximation of fixing numerical attributes in databases under integrity constraints. Inf. Syst. 33(4-5), 407–434 (2008)
31. Meliou, A., Gatterbauer, W., Halpern, J.Y., Koch, C., Moore, K.F., Suciu, D.: Causality in databases. IEEE Data Eng. Bull. 33(3), 59–67 (2010)
32. de Calmès, M., Dubois, D., Hüllermeier, E., Prade, H., Sedes, F.: Flexibility and fuzzy case-based evaluation in querying: An illustration in an experimental setting. Inter. J. of Uncertainty, Fuzziness and Knowledge-Based Systems 11(1), 43–66 (2003)

oarieli@mta.ac.il

Learning the News in Social Networks

Krishnan Rajagopalan[1], Venkatesh Srinivasan[2], and Alex Thomo[3]

[1] Digital Media Technologies, Motion Picture Association of America, Los Angeles, CA, USA
krishnan_rajagopalan@yahoo.com
[2] Department of Computer Science, University of Victoria, Victoria, BC, Canada
venkat@cs.uvic.ca
[3] Department of Computer Science, University of Victoria, Victoria, BC, Canada
thomo@cs.uvic.ca

Abstract. In social media such as facebook, the most popular desire is to learn the news about other people. In this paper, we study the following problem related to information propagation: Suppose that there is a set U of N users in a social network. They meet online from time to time and share information they know about themselves and the other users in the network. Whenever a group $g \subset U$ of users meet, they want to know who has the latest information about every user in U. A naive solution to this problem is to use timestamps. However, there are drawbacks to this scheme including the burden on the users to maintain reliable timestamps and the fact that the timestamps grow unbounded over time. It is natural to ask if it is possible to learn the latest information without using timestamps. We present an efficient method which removes the need to timestamp user information (news). Instead, only the meetings of the groups have to be indexed. Furthermore, we show that this indexing can be performed using a finite set of labels so that each user stores at most $O(N^2 \log N)$ bits of information. We also show that this bound can be improved in some cases if we have further information on the topology of the network.

1 Introduction

Today, it is hard to find people, at least from certain demographics, who are not part of a social network, such as Facebook, Twitter, LinkedIn, and so on. One of the main reasons for joining a social network is to satisfy the inherent human need of learning "the news" about other people. In virtually all the social networks, there are implicit or explicit groups formed for the sole purpose of disseminating information. However, this bears the question of how to determine the latest information about users in a network.

The setting we study in this paper is as follows. Suppose that there is a set U of N users in a social network. They meet online in groups from time to time and share information they know about themselves and other users in the network. Whenever a group $g \subset U$ of users meet, they want to know who has the latest information about every user in U. We assume honest users who are

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 298–311, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

willing to share all they know with the other members of the group. A user can be a member of more than one group, and thus serve as a "bridge" for news propagation. Also we assume that during a group meeting each member of the group gets up to date with respect to all the users that the other group members have information (news) for. Therefore, for a given user $u \in U$, the members of a group g, need to determine who (in the group) has the latest information about u.

A naive solution to this problem is to use timestamps. However, there are two problems with letting users timestamp their personal information. First, there is the burden on each user to produce and use reliable timestamps conforming to some standard. Second, there is an additional problem that the label set for such timestamps grows unbounded over time. Can we avoid this?

In this paper, we present an efficient method to infer the latest information without using timestamps for the user information. We present an algorithm which is able to determine who has the latest information for a given user by reasoning about a "latest information graph" structure we introduce. In this algorithm, only the meetings of the groups have to be indexed (numbered). These indexes can be considered as a form of rudimentary timestamps, but they are on the group meetings level, not on the personal user information level.

Our Contributions. More concretely, we make the following contributions in this paper.

- We introduce the notion of the "latest information graph" that allows users present in a meeting to compare and choose the latest information they have about any other user in the network.
- For a network of N users with no restriction on the network topology, we show that it is sufficient for each user to maintain $O(N^2 \log N)$ bits of information improving a previous bound of $O(N^3 \log N)$[9].
- For the special case of mesh topology, the n-dimensional hypercube, we show that it is sufficient to maintain $O(\log^3 N)$ bits of information for $N = 2^n$ users. This is a significant improvement over the general case.

Related Work. "Gossiping" is a well studied problem in distributed systems [6]. In the traditional setting of this problem, a user has a piece of information that needs to be communicated to all the other users and the goal is to design a protocol that minimizes the total communication needed for a fixed network topology.

Bounded time-stamps have been studied for a long time in various models of distributed systems. Israeli and Li introduced this notion [7] for shared memory model. Their results were further improved in [3] and [4]. However, we study this problem for social networks that can be viewed as an asynchronous communication model and their results are not comparable to ours. Another direction of research studies this problem in the model of asynchronous cellular automata that starts with a different assumptions for interaction (See [1], [2]).

Very recently, Lind *et al.* [8] use a simple model to understand information propagation in scale-free and small-world networks and compare their results

oarieli@mta.ac.il

to real empirical network of social acquaintances. Fan *et al.* [5] point out the unreliability of time-stamps in the setting of relational databases and the need to maintain data currency in the absence of timestamps.

Organization. In Section 2, we formally define our setting and give a characterization lemma about latest information. Sections 3, 4 and 5 explain how the users compare and update their latest information during meetings. We outline our algorithm and its analysis is Section 6. In Section 7, we prove our result on the special case of hypercubes. We end with our conclusions in Section 8.

2 Groups, Meetings, Information

We denote by U the set of users in a social network and by G the set of groups the users have created over time. Each group has a name, and for simplicity, we will blur the distinction between a name and the group it names. There are possibly many group meetings, and in each such occasion, the members of the group exchange the latest information about themselves and other users they know about in the network. We assume that a user, who belongs in more than one group, does not participate in more than one meeting at the same time. Also we assume that a group meeting can happen only if all the users are available and willing to participate.

If two groups g, h have some user in common, we say that they are *bridged*. Given a group $g \in G$, we denote its meetings by $g_1, g_2, \ldots$. Consider a sequence σ of meetings of different groups (e.g. $g_1h_1g_2k_1h_2$). We say that a meeting h_j *depends on* a meeting g_i, and write $g_i \prec h_j$, if groups g and h are bridged, and g_i happened before h_j in time. We denote by $\preceq^*$ the reflexive and transitive closure of $\prec$.

Latest Information. Given a sequence σ of meetings, and a user u, let g_i be the last meeting that u participates in. Let σ_u be the subsequence of σ containing only meetings which are "smaller" ($\preceq^*$) than g_i. Given another user t, let e_j be the last meeting in σ_u that t participates in (of course $t \in e_j$). Clearly, the information that u has about t corresponds to this meeting e_j. We denote g_i by $last_u(\sigma)$ and e_j by $last_{u,t}(\sigma)$.

For any three users u, v, and t, we have that $last_{u,t}(\sigma)$ can be put in a $\preceq^*$ relationship with $last_{v,t}(\sigma)$. This is because both $last_{u,t}(\sigma)$ and $last_{v,t}(\sigma)$ are meetings where t participates. In general the meetings where a user t participates are called *t-meetings*. Any two t-meetings can be compared with respect to $\preceq^*$. Therefore, $\preceq^*$ is a total order for t-meetings. Purely for convenience of notation, we assume that there is an initial meeting labeled 0 involving all the users.

Figure 1 shows a communication sequence $\sigma = f_1g_1h_1g_2$ of four meetings. There are four users u, v, w and x. Note that $last_u(\sigma) = g_2$ and $last_{u,w}(\sigma) = h_1$. Furthermore, $\sigma_x = f_1g_1h_1$ while $\sigma_u = f_1g_1h_1g_2$. $g_1 \preceq h_1$ as they have a user v in common. Also note that $f_1 \preceq^* g_2$ because $f_1 \preceq h_1$ and $h_1 \preceq g_2$.

oarieli@mta.ac.il

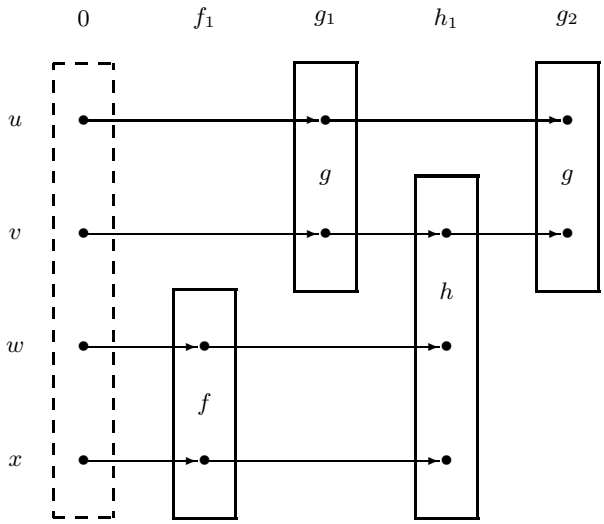


Fig. 1. An example

Lemma 1. *Let u, v, and t be three users in the network, and let σ be a communication sequence. Then one of the following holds.*

(i) $last_{u,t}(\sigma)$, $last_{v,t}(\sigma) \in \sigma_u \cap \sigma_v$ *and* $last_{u,t}(\sigma) = last_{v,t}(\sigma)$.
(ii) $last_{u,t}(\sigma) \in \sigma_u \cap \sigma_v$, $last_{v,t}(\sigma) \in \sigma_v \setminus \sigma_u$ *and* $last_{u,t}(\sigma) \preceq^* last_{v,t}(\sigma)$.
(iii) $last_{v,t}(\sigma) \in \sigma_u \cap \sigma_v$, $last_{u,t}(\sigma) \in \sigma_u \setminus \sigma_v$ *and* $last_{v,t}(\sigma) \preceq^* last_{u,t}(\sigma)$.

Proof. **Case 1.** Since $last_{u,t}(\sigma)$, $last_{v,t}(\sigma)$ are t-meetings, and both in σ_v, we have $last_{u,t}(\sigma) \preceq^* last_{v,t}(\sigma)$. By a symmetric argument, substituting σ_u for σ_v, we get $last_{v,t}(\sigma) \preceq^* last_{u,t}(\sigma)$. Therefore, $last_{u,t}(\sigma) = last_{v,t}(\sigma)$.

Case 2. One of $last_{u,t}(\sigma)$ or $last_{v,t}(\sigma)$ belongs to $\sigma_u \cap \sigma_v$ and the other does not belong to $\sigma_u \cap \sigma_v$. Suppose $last_{u,t}(\sigma) \in \sigma_u \cap \sigma_v$ and $last_{v,t}(\sigma) \in \sigma_v \setminus \sigma_u$. Then, since $last_{u,t}(\sigma) \in \sigma_v$ and we have $last_{u,t}(\sigma) \preceq^* last_{v,t}(\sigma)$. A similar argument holds for the other possibility.

Case 3. Let, if possible, $last_{u,t}(\sigma) \in \sigma_u \setminus \sigma_v$ and $last_{v,t}(\sigma) \in \sigma_v \setminus \sigma_u$. Since, $last_{u,t}(\sigma)$ and $last_{v,t}(\sigma)$ are both t-meetings, they must be totally ordered, i.e, either (a) $last_{u,t}(\sigma) \preceq^* last_{v,t}(\sigma)$ or (b) $last_{v,t}(\sigma) \preceq^* last_{u,t}(\sigma)$. Suppose we have (a). This means that $last_{u,t}(\sigma) \in \sigma_v$ because σ_v is the subsequence of σ containing all the meetings that are $\preceq^*$ than $last_{v,t}(\sigma)$. $last_{u,t}(\sigma) \in \sigma_v$ is a contradiction. A similar argument holds for the other possibility. □

oarieli@mta.ac.il

3 Latest Information Graph

The latest information that a user u has at the end of a meetings sequence σ can be organized as a graph $\mathcal{G}_u(\sigma) = (V_u(\sigma), E_u(\sigma))$, where

$$V_u(\sigma) = \{(t, last_{u,t}(\sigma)) : t \in U\}$$
$$E_u(\sigma) = \{[(t, last_{u,t}(\sigma)), (t', last_{u,t'}(\sigma))] : last_{u,t}(\sigma) \preceq^* last_{u,t'}(\sigma)\}.$$

Since $\preceq^*$ is reflexive, a self-loop edge is also contained in $E_u(\sigma)$ for each node in $V_u(\sigma)$.

Whenever two users u and v meet in some group meeting, they want to decide who has the latest information about some other user t. For this we present the following lemma.

Lemma 2. *$last_{u,t}(\sigma) \preceq^* last_{v,t}(\sigma)$, if and only if, there exists $t', t'' \in U$, such that*

$$last_{u,t'}(\sigma) = last_{v,t''}(\sigma)$$
$$[(t, last_{u,t}(\sigma)), (t', last_{u,t'}(\sigma))] \in E_u(\sigma)$$

Proof. ($\Rightarrow$). $last_{u,t}(\sigma) \preceq^* last_{v,t}(\sigma)$. We have the following possibilities.

Case 1. $last_{u,t}(\sigma) = last_{v,t}(\sigma)$. From Lemma 1, we have, $last_{u,t}(\sigma), last_{v,t}(\sigma) \in \sigma_u \cap \sigma_v$, which implies

$$(t, last_{u,t}(\sigma)) \in V_u(\sigma) \cap V_v(\sigma).$$

Since we always have self-loops in information graphs, we have $[(t, last_{u,t}(\sigma)), (t, last_{u,t}(\sigma))] \in E_u(\sigma)$. We set now $t' = t$ and the claim follows.

Case 2. $last_{u,t}(\sigma) \neq last_{v,t}(\sigma)$. From Lemma 1, we have $last_{u,t}(\sigma) \in \sigma_u \cap \sigma_v$ and $last_{v,t}(\sigma) \in \sigma_v \setminus \sigma_u$. Let t' be a user who participates in the $\preceq^*$-greatest meeting in $\sigma_u \cap \sigma_v$. Clearly, $last_{v,t}(\sigma) \preceq^* last_{u,t'}(\sigma)$. Thus, we have

$$[(t, last_{u,t}(\sigma)), (t', last_{u,t'}(\sigma))] \in E_u(\sigma).$$

Also, it is clear that $(t', last_{v,t'}(\sigma)) \in V_v(\sigma)$.

($\Leftarrow$). To prove this direction, we observe that $last_{u,t}(\sigma) \preceq^* last_{u,t'}(\sigma)$ and $last_{u,t'}(\sigma) \in \sigma_u \cap \sigma_v$. Therefore, $last_{u,t}(\sigma)$ also belongs to $\sigma_u \cap \sigma_v$. Now it follows from Lemma 1 that $last_{u,t}(\sigma) \preceq^* last_{v,t}(\sigma)$. From these facts the claim follows. □

4 Updating Latest Information

Let $\mathcal{G}_u(\sigma') = (V_u(\sigma'), E_u(\sigma'))$ be the latest information graph for u and $\mathcal{G}_v(\sigma') = (V_v(\sigma'), E_v(\sigma'))$ be the latest information graph for v at the end of communication sequence σ'. After a meeting h involving two users u and v, each of them update their latest information for the communication sequence $\sigma = \sigma' h$ by constructing a common updated graph $\mathcal{G}_{uv}(\sigma) = (V_{uv}(\sigma), E_{uv}(\sigma))$ as follows :

oarieli@mta.ac.il

(i) Both $V_{uv}(\sigma)$ and $E_{uv}(\sigma)$ are set to $\emptyset$.
(ii) Let the new meeting under which u and v have synchronized be labeled by l. The process by which a new label l is assigned to the meeting is explained later.
(iii) $(u, l), (v, l)$ are added to $V_{uv}(\sigma)$.
(iv) For each $w \in U - \{u, v\}$, the pair (w, e) is added to $V_{uv}(\sigma)$, where e is the latest of what u and v know about w(as given in lemma 2).
(v) For $s \in \{u, v\}$ and for each $w \in U - \{u, v\}$, $((w, e), (s, l))$ is added to $E_{uv}(\sigma)$.
(vi) For all s, t belonging to $U - \{u, v\}$, $((s, e), (t, f))$ is added to $E_{uv}(\sigma)$ if there exists $r, r' \in U$ s.t $((r, e), (r', f)) \in E_u(\sigma') \cup E_v(\sigma')$.

It can be easily seen that the update procedure ensures that each user maintains not only the latest information about all other users, but also maintains any ordering that exists between these meetings.

From Lemma 2, we know that, to compare latest information, we only need to look at the meetings which are currently in the latest information graph of each user. Recall that the vertices of $\mathcal{G}_u$ and $\mathcal{G}_v$ are identified by *labels*. If a meeting e occurs in both $\mathcal{G}_u$ and $\mathcal{G}_v$, then it must be given the same label in both the graphs. Since the update procedure labels only the current meeting, keeping the other meeting labels unchanged, this is ensured. So far, we have achieved our goal of indexing meetings rather than timestamping user information.

5 Achieving Boundedness of Label Sets

Now, a new label (index) has to be chosen for the current meeting. It must be ensured that the label assigned to this meeting is consistent across the system—i.e., if the same label appears in the current latest information graph of different users, the corresponding meeting is actually the same.

Unfortunately, the users in a meeting g_i cannot directly see all the g-meetings which belong to the latest information graphs of the entire system. A g-meeting e may be part of the latest information of user *outside* g.

To enable the users in g to know about all g-meetings in $\{\mathcal{G}_u(\sigma)\}_{u \in U}$, we will maintain auxiliary information.

Auxiliary information. The *auxiliary information* of user u after σ, $auxiliary_u(\sigma)$, is a set of meetings of the form $last_{v,w}(\sigma_{last_{u,v}(\sigma)})$ for some $v, w \in U$. This is the latest w-meeting which v knows about up to the meeting $last_{u,v}(\sigma)$. We abbreviate $last_{v,w}(\sigma_{last_{u,v}(\sigma)})$ by $last_{u,v,w}(\sigma)$.

We represent each auxiliary meeting $e = last_{u,v,w}(\sigma)$ in $auxiliary_u(\sigma)$ as a quadruple (u, v, w, e). However, we can think of the auxiliary information as a set of meetings. Then, $e \in auxiliary_u(\sigma)$ will indicate that for some $v, w \in U$, $(u, v, w, e) \in auxiliary_u(\sigma)$.

Lemma 3 ([9]). *Let σ be a communication sequence, $u \in U$ and e a u-meeting in σ. If $e \notin auxiliary_u(\sigma)$, then e is not a meeting in $\mathcal{G}_w(\sigma)$ for any user $w \in U$.*

oarieli@mta.ac.il

So, a user u can keep track of which of its labels are "in use" in the system by maintaining auxiliary information. Each u-meeting e initially belongs to $\mathcal{G}_u(\sigma)$, and hence to $auxiliary_u(\sigma)$ as well. As the computation progresses, e gradually "recedes" into the background and disappears from the latest information graphs of the system. Eventually, when e disappears from $auxiliary_u(\sigma)$, u can be sure that e no longer belongs to $\mathcal{G}_w(\sigma)$ for any user $w \in U$.

Since $auxiliary_u(\sigma)$ is a bounded set, u knows that only finitely many of its labels are in use at any given time. So, by using a sufficiently large finite set of labels, each new meeting can always be assigned an unambiguous label by the users which take part in the meeting.

6 Our Algorithm

For convenience, the algorithm starts with an initial meeting involving all the users. This meeting is denoted by $\langle \mathcal{P}, \ell_0 \rangle$ for an arbitrary but fixed label $\ell_0 \in \mathcal{L}$.

Subsequently, for any meeting of a group g, the users in g do the following:

(i) When a new g-labelled meeting e occurs, the users in g assign a label $\langle g, \ell \rangle$ to e which does not appear in $auxiliary_u(\sigma)$ for any user u in g. Lemma 3 guarantees that this new label does not appear in $\mathcal{G}_w(\sigma)$ for any user $w \in U$. Let $N = |U|$. Since each user keeps track of N^2 auxiliary meetings and at most N users can be at a meeting, there need be only N^3 labels in $\mathcal{L}$.

(ii) The users participating in e now compare and update their latest information about each user $w \notin e$ by checking labels of meetings across their latest information graphs as described in Lemma 2 and the update procedure outlined in the previous section.

(iii) Auxiliary information can be locally updated once the users have decided who has the best latest information—if $u, v \in g$ and $last_{u,w}(\sigma)$ is better than $last_{v,w}(\sigma)$ for $w \in U$, then any auxiliary information of the form $last_{u,w,w'}(\sigma)$ must necessarily be better than the corresponding information $last_{v,w,w'}(\sigma)$, for $w' \in U$.

The Amount of Local Information.

Theorem 1. *Each user $u \in U$ needs to maintain at most $O(N^2 \log N)$ bits of information, where $N = |U|$.*

Proof. Each new meeting e is assigned a label of the form $\langle g, \ell \rangle$, where g was the group of users that participated in e and $\ell \in \mathcal{L}$.

To write down $g \subseteq U$, we need, in general, N bits. This component of the label is required to guarantee that all auxiliary meetings in the system have distinct labels, since the set $\mathcal{L}$ is common across all users. However, we do not really need to use all of g in the label for e to ensure this property. If we order U as $\{u_1, u_2, \ldots, u_N\}$, it suffices to label e by $\langle u_i, \ell \rangle$ where, among the users in g, u_i has the least index with respect to our ordering of U.

oarieli@mta.ac.il

Thus, we can modify our automaton so that the users label each meeting by a pair $\langle u, \ell \rangle$, where $u \in U$ and $\ell \in \mathcal{L}$. This pair can be written down using $O(\log N)$ bits.

The *latest information* of each user is stored in two parts:

- An array which stores for each user $u \in U$, the latest it knows about u. This requires $N \log N$ bits.
- The edges of the latest information graph are stored in the form of an adjacency matrix which requires N^2 bits.

The *auxiliary information* of each user clearly requires $O(N^2 \log N)$ bits. Hence, the total information can be described in $O(N^2 \log N)$ bits.

In the following section, we show that when we consider particular case of mesh network topology, the hypercube, we can improve the bound even further. We believe that the hypercube is a good representative of a network with "large" groups. A good understanding of this problem for hypercube is relevant to study of social networks with large community sizes.

7 Hypercube

An *n-dimensional hypercube*, $\mathcal{H}_n$, is a network of 2^n users. The users are numbered 0 through $2^n - 1$ and each user is identified by its binary representation. Each face of the hypercube represents a group of the users that meet from time to time. That is, exactly those users which agree on the kth bit, $1 \leq k \leq n$, can form a group. Thus each group involves 2^{n-1} users and there are $2n$ possible groups. We will call the groups $g_1, g_2, \ldots, g_n, g_{n+1}, \ldots, g_{2n}$, with $g_i,\ i \in \{1, \ldots, n\}$, denoting a group of all the users that have a 1 in their i^{th} bit and $g_j,\ j \in \{n+1, \ldots, 2n\}$, denoting a group of all the users that have a 0 in their $(j-n)^{th}$ bit.

We call two groups g_i and g_j *complementary* if either $i = j - n$ or $j = i - n$. This means that the set of the users that take part in g_i and g_j are disjoint and together make up the set of all users in the system.

Note that two different g-meetings cannot be at the same time in the latest information graph of any user during the computation. The following lemma proves this.

Lemma 4. *Let σ be a communication sequence. For any $u, v, w \in U$, let e be a g-meeting and $e = last_{u,v}(\sigma)$. Let $f = last_{u,w}(\sigma)$. Then, f cannot be a g-meeting different from e.*

Proof. Suppose, f is a g-meeting and f is different from e. Then either $e \preceq^+ f$ or $f \preceq^+ e$, since all g-meetings are totally ordered.

Without loss of generality, let $e \preceq^+ f$. Then, v is a user that participated in f and $e \preceq^+ f$. Hence $e \neq last_{u,v}(\sigma)$ which is a contradiction.

A similar argument hold for the other possibility.

oarieli@mta.ac.il

In the case of the hypercube, the number of possible groups is just $2n$ and there are 2^n users. It follows that the latest information graph in general is likely to have many copies of the same g-meeting for many groups g. So, can we "collapse" all the occurrences of this meeting and maintain just one vertex corresponding to that meeting in the graph? This will reduce the number of vertices in the latest information graph from $O(2^n)$ to $O(n)$. This motivates the following definition for the latest information graph of a hypercube.

Latest Information Graph. Let σ be a communication sequence and $u, v \in U$. The *latest information graph* of u after σ is a directed graph $\mathcal{G}_u(\sigma) = (V_u(\sigma), E_u(\sigma))$ where

$$V_u(\sigma) = \{e \mid \exists v \in \mathcal{P} \ s.t \ e \ = \ last_{u,v}(\sigma)\}$$

$$E_u(\sigma) = \{(e, f) \mid e \preceq^* f\}$$

That is, the latest information graph is a directed graph in which each vertex has a label of a meeting that corresponds to the latest information that u knows about some other user in U. There is an edge from vertex e to f if e is below f. Note that there are at most $2n$ vertices in this graph.

The latest information that any user u knows about any other user u' in the system can be recovered from the latest information graph of u by finding all the meetings that u' participates in and getting the $\preceq^* -$ maximal meeting from them. We only have to ensure that there exists a proper update procedure to keep updating the graph after each meeting. The following section shows how to maintain and update the latest information graph.

Updating Latest Information. As in the general case, we outline the procedure by which two users can update their latest information when they meet. When a meeting takes place, the users update their information pairwise, and finally propagate the common updated graph to all the users that participated in the current meeting. The peculiarity of the hypercube architecture is that either two meetings are ordered or are complementary to each other. This fact results in a simple update procedure which is outlined below.

Let $\mathcal{G}_u(\sigma') = (V_u(\sigma'), E_u(\sigma'))$ be the old graph for u, $\mathcal{G}_v(\sigma') = (V_v(\sigma'), E_v(\sigma'))$ be the old graph for v, and let $\mathcal{G}_{uv}(\sigma) = (V_{uv}(\sigma), E_{uv}(\sigma))$ denote the common updated graph for u and v where $\sigma = \sigma' g$. When two users u and v meet during a g-meeting, each of them update their latest information by constructing the new graph $\mathcal{G}_{uv}(\sigma)$ as follows :

(i) A new label l is assigned to the current meeting g.
(ii) If either $last_u(\sigma')$ or $last_v(\sigma')$ belongs to the set $S = V_u(\sigma') \cap V_v(\sigma')$:
If $last_u(\sigma') \in$ S:
Let $\mathcal{G}'_{uv}(\sigma) = (V'_{uv}(\sigma), E'_{uv}(\sigma))$ be the initial updated graph. Then,

$$V'_{uv}(\sigma) \ = \ V_q(\sigma') \cup \ l.$$

$$E'_{uv}(\sigma) \ = \ E_q(\sigma') \ \cup \ \{(e, l) \mid e \in {}_{V_q}(\sigma')\}.$$

oarieli@mta.ac.il

We then remove the redundant meetings in $\mathcal{G}'_{uv}(\sigma)$ to get the final updated graph. We say a meeting e is *redundant* in $\mathcal{G}'_{uv}(\sigma)$ if for all u' taking part in e, $e \preceq^+ last_{v,u'}(\sigma)$. Let the set of redundant meetings be denoted by $R_{uv}(\sigma)$. Then,

$$V_{uv}(\sigma) \;=\; V'_{uv}(\sigma) \;-\; R_{uv}(\sigma).$$

If $last_u(\sigma) \in$ S: Do the same with the roles of u and v reversed.

(iii) If neither $last_u(\sigma)$ nor $last_v(\sigma)$ belong to the set S:
Then,

$$V_{uv}(\sigma) \;=\{\; last_u(\sigma) \;\cup\; last_v(\sigma) \;\cup\; l\}.$$

$$E_{uv}(\sigma) \;=\; \{(last_u(\sigma), l), (last_v(\sigma), l)\}.$$

The following lemmas show that the update procedure works correctly.

Lemma 5. *Let $u, v \in U$ and σ be a communication sequence. Consider the set $S = V_u(\sigma) \cap V_v(\sigma)$. Suppose $last_u(\sigma) \in S$. Then, for any $u' \in U$,*

$$e \;=\; last_{u,u'}(\sigma) \quad \preceq^* \quad f \;=\; last_{v,u'}(\sigma).$$

Proof. In $\mathcal{G}_u(\sigma)$, $e \preceq^* last_u(\sigma)$. Also, $last_u(\sigma) \in \mathcal{G}_v(\sigma)$ and $f \;=\; last_{v,u'}(\sigma)$. Hence, $e \preceq^* f$ since all u'-meetings are totally ordered.

Lemma 6. *Let $u, v \in U$ and σ be a communication sequence. Consider the set $S = V_u(\sigma) \cap V_v(\sigma)$. Suppose both $last_u(\sigma)$ and $last_v(\sigma)$ do not belong to S. Then,*

- *For all u' that participated in $last_u(\sigma)$, $last_{v,u'}(\sigma) \preceq^* last_{u,u'}(\sigma)$.*
- *For all u'' that participated in $last_v(\sigma)$, $last_{u,u''}(\sigma) \preceq^* last_{v,u''}(\sigma)$.*

Proof. Suppose that there exists a user u' that participated in $last_u(\sigma)$ such that $last_{u,u'}(\sigma) \preceq^* last_{v,u'}(\sigma)$. Then, $last_{u,u'}(\sigma) \in \sigma_u \cap \sigma_v$. Hence, $last_u(\sigma) \in S$, which is a contradiction. A symmetric argument holds for the other part.

So, whenever two users meet, they can easily compare and update their latest information provided a new label corresponding to the current meeting can be chosen. As in the general case, this is done by looking at the auxiliary information of users.

Eliminating Redundant Meetings Efficiently. We now show how to remove redundant meetings much more efficiently than exhaustive search.

Lemma 7. *Let σ be a communication sequence. Let $u, v \in U$ and $e \in V'_{uv}(\sigma)$. e is* redundant *iff $\exists f, f' \in V'_{uv}$ such that f and f' are complementary meetings, $e \preceq^* f$ and $e \preceq^* f'$.*

Proof. ($\Longleftarrow$) Since f and f' are complementary meetings, and each involves 2^{n-1} users, they together cover all the users in U. Therefore, since $e \preceq^* f$ and $e \preceq^* f'$, there does not exist any user $u' \in U$ such that e is the latest that u (or v) know about u'. Hence e is redundant.

oarieli@mta.ac.il

($\Longrightarrow$) Let e be a redundant meeting. Suppose e is a g_k-meeting, $k \leq n$, i.e, e is a meeting of all the users that have a 1 in their k^{th} bit.

Suppose there do not exist meetings f and f' such that they are complementary and $e \preceq^* f$ and $e \preceq^* f'$. Then, there can be at most $n-1$ meetings above e in V'_{uv} and each of them is of the form $g_i,\ i \neq k, k+n$. Let T be the set of these meetings. Now consider the user $w \in U$ which has a bit representation as follows :

- A 1 in the kth bit.
- For $i \leq n,\ i \neq k$, a 0 in the ith bit if T contains a g_i-meeting, 1 if T contains a g_{i+n}-meeting and either 0 or 1 in the other bits not covered by meetings in T.

It follows that for the user $w,\ e$ is the latest meeting that u or v know about w, which is a contradiction.

A similar argument holds if e is a g_k-meeting, $k > n$.

So, in order to check if a meeting is redundant, it is sufficient to check if this meeting is below two complementary meetings. This is much more efficient than exhaustively checking if e is not the latest meeting for every user in e.

Auxiliary Information. Let σ be a communication sequence and $u \in U$. The *auxiliary information* of u after σ, $auxiliary_u(\sigma)$, is a directed graph $\mathcal{G}^s_u(\sigma) = (V^s_u(\sigma), E^s_u(\sigma))$ where

$$V^s_u(\sigma) = \{e \mid \exists u', u'' \in U \ s.t \ e \ = \ last_{u,u',u''}(\sigma)\}$$

$E^s_u(\sigma) = \{(e, f) \mid e \preceq^* f \ and \ \exists \ u', u'', u''' \in U \ s.t \ e = last_{u,u',u''}(\sigma) \ and \ f = last_{u,u',u'''}(\sigma)\}$.

That is, the auxiliary information maintains, for each of the meetings e in the latest information, the set of meetings which gives the latest information that users in e know about every other user in the system. This is maintained as a directed graph which has at most $2n$ maximal components, each of which has at most $2n$ vertices.

Updating Auxiliary Information. Auxiliary information can be updated as and when the latest information is being updated. Recall that when two users u and v meet, there are the two possibilities corresponding to which the auxiliary information is updated as given below :

Let σ be a communication sequence and let $\mathcal{G}_u(\sigma')$ and $\mathcal{G}_v(\sigma')$ denote the initial latest information graphs of u and v respectively, i.e, before the occurrence of the current meeting g. Let $\sigma = \sigma' g$. We denote by $\mathcal{G}_{uv}(\sigma)$, the common updated latest information graph of u and v after the meeting g. Let $\mathcal{G}^s_u(\sigma')$ be the auxiliary information of u and $\mathcal{G}^s_v(\sigma')$ the auxiliary information of v. We denote by $\mathcal{G}^s_{uv}(\sigma)$, the common updated auxiliary graph of u and v. Now, when the u and v meet, the possibilities are :

oarieli@mta.ac.il

- **Case 1 :** One of $last_u(\sigma')$ or $last_v(\sigma')$ belongs to the set S.
  Suppose $last_u(\sigma') \in$ S.
  Let $\mathcal{G}'$ be the union of all those *maximal components* of $\mathcal{G}_q^s(\sigma')$ such that the maximal meeting in that component belongs to R_{uv}. Then, we get the common updated auxiliary graph as follows:

$$\mathcal{G}_{uv}^s(\sigma) \ = \ (\ \mathcal{G}_q^s(\sigma') \ - \ \mathcal{G}' \ ) \ \cup \ \mathcal{G}_{uv}(\sigma).$$

  A symmetric argument holds for the other possibility.
- **Case 2 :** Neither $last_u(\sigma')$ nor $last_v(\sigma')$ belong to the set S.
  Then, $\mathcal{G}_{uv}^s(\sigma) \ = \ \mathcal{G}_u(\sigma') \ \cup \ \mathcal{G}_v(\sigma') \ \cup \ (\mathcal{G} = (V, E))$
  where $V = \{last_u(\sigma'), last_v(\sigma'), l\}$ and
  $E = \{(last_v(\sigma'), l), (last_v(\sigma'), l)\}$. Remove redundant meetings from $\mathcal{G}_{uv}^s(\sigma)$, if any.

In the first case, the above procedure works since if $last_u(\sigma')$ belongs to S, then the auxiliary information of v is at least as good as the auxiliary information of u.

In the second case, we have observed that the common updated latest information graph has just three vertices : $last_u(\sigma')$, $last_v(\sigma')$ and l. Since $last_u(\sigma')$ and $last_v(\sigma')$ are complementary meetings, for all the users u' that participated in g,

$$\{last_{v,u',u''}(\sigma) \mid r \ = u \ or \ v \ and \ u'' \ \in U\} \subseteq \{last_u(\sigma'), last_v(\sigma')\}.$$

For any user u' that did not participate in g (there are 2^{n-1} such users), since the latest that u or v know about u' is either $last_u(\sigma')$ or $last_v(\sigma')$, we have,

$$\{last_{r,u',u''}(\sigma) \mid r \ = u \ or \ v \ and \ u'' \ \in U\} \subseteq \{e \mid e \in V_u \ or \ e \in V_v\}.$$

So, we have shown that the update procedure for auxiliary information outlined earlier works.

The Amount of Local Information.

Theorem 2. *Each user $u \in \mathcal{H}_n$ needs to maintain at most $O(n^3)$ bits of information.*

Proof. The local information for user u consists of its latest and auxiliary information. We estimate the number of bits required to store this.

The *latest information* of each user is stored in two parts:

- An $2n \times 1$ array which stores the labels of at most $2n$ meeting containing latest information. Since we have shown, for the general case, that it suffices to have N^3 labels in the system, each label requires $O(logN) \ = \ O(n)$ bits. This component of the latest information therefore requires $O(n^2)$ bits.
- A $2n \times 2n$ adjacency matrix which stores the edges of the latest information graph. This requires $O(n^2)$ bits.

oarieli@mta.ac.il

Therefore, storing the latest information requires $O(n^2)$ bits in all.

The *auxiliary information* of each user is stored in two parts:

- An $2n \times 2n$ array which stores the labels of at most $4n^2$ auxiliary meetings. This component of the auxiliary information therefore requires $O(n^3)$ bits.
- $2n$ adjacency matrices of dimension $2n \times 2n$ which store the edges of the auxiliary graph. This clearly requires $O(n^3)$ bits.

Therefore, the auxiliary information requires $O(n^3)$ bits in all. Hence, local information of each users requires $O(n^3)$ bits.

8 Conclusions

In this paper, we have studied a natural problem related to information dissemination in social networks in which users meet in groups over time and are interested in learning the latest news known to the users present in the meeting about all the other users in the network.

Identifying the user in the meeting with the latest information becomes tricky if the users are only allowed to label each meeting with a string from a finite, bounded set of labels. The main difficulty is to ensure that it is still possible to compare and choose among two labels while the labels are being recycled over time.

We have shown two interesting results:

- For a network of N users with no restriction on the network topology, we have shown that it is sufficient for each user to maintain $O(N^2 \log N)$ bits of information improving the previous bound of $O(N^3 \log N)$[9].
- For the special case of mesh topology, the n-dimensional hypercube, we have shown that is sufficient to maintain $O(n^3)$ bits of information for $N = 2^n$ users. This is a significant improvement over the general case.

For future research, we would like to try and extend our results to other topologies of interest. In particular, we would like to understand the structure and connectivity proerties of communities in social networks to see what further assumptions can be made in the case of a scale-free network.

Acknowledgements. We would like to thank Madhavan Mukund and Milind Sohoni for introducing us to this problem and for many useful discussions.

References

1. Cori, R., Metivier, Y.: Asynchronous Mappings and Asynchronous Cellular Automata. Information and Computation 106, 159–202 (1993)
2. Cori, R., Sopena, E.: Some Combinatorial Aspects of Time-stamp Systems. European Journal of Combinatorics 14, 95–102 (1993)

oarieli@mta.ac.il

3. Dolev, D., Shavit, N.: Bounded Concurrent Time-stamps are Constructible. In: Proc. of 21th Annual ACM Symposium on Theory of Computing, pp. 454–466 (1989)
4. Dwork, C., Waarts, O.: Simple and Efficient Bounded Concurrent Timestamping or Bounded Concurrent Time-stamps are Comprehensible. In: Proc. of 24th Annual ACM Symposium on Theory of Computing, pp. 655–666 (1992)
5. Fan, W., Geerts, F., Wijsen, J.: Determining the Currency of Data. In: Proc. of the 30th ACM Symposium on Principles of Database Systems, pp. 71–82 (2011)
6. Hedetniemi, S.M., Hedetniemi, S.T., Liestman, A.L.: A Survey of Gossiping and Broadcasting in Communication Networks. Networks 18, 319–349 (1988)
7. Israeli, A., Li, M.: Bounded time-stamps. In: Proc. of 28th IEEE Conference on Foundations of Computer Science, pp. 371–382 (1987)
8. Lind, P.G., da Silva, L.R., Andrade Jr, J.S., Herrmann, H.J.: Spreading Gossip in Social Networks. Phys. Rev. E 76(3) (2007)
9. Mukund, M., Sohoni, M.: Keeping Track of the Latest Gossip in a Distributed System. Distributed Computing 10(3), 137–148 (1997)

oarieli@mta.ac.il

Verifying Resource Requirements for Ontology-Driven Rule-Based Agents

Abdur Rakib, Rokan Uddin Faruqui, and Wendy MacCaull

StFX Centre for Logic and Information
St. Francis Xavier University, Canada
{arakib,x2010mcd,wmaccaul}@stfx.ca

Abstract. Recent efforts towards the Semantic Web have resulted in powerful languages such as Semantic Web Rule Language (SWRL) based on OWL-DL and RuleML. Rule languages and inference engines incorporate reasoning capabilities to Semantic Web application systems. In this paper we present an approach for the design and specification of ontology-driven multi-agent rule-based systems. We use the Maude rewriting system and its Linear Temporal Logic (LTL) model checking tool to verify response time guarantees for the target systems. We present TOVRBA, an extended version of a verification tool developed by the first author, for ontology-driven multi-agent rule-based systems which allows the designer to specify information about agents' interactions, behavior, and execution strategies at different levels of abstraction. TOVRBA generates an encoding of the system for the Maude LTL model checker, allowing properties of the system to be verified. We illustrate the use of the framework on a simple healthcare system.

Keywords: Multi-agent systems, Semantic Web, Ontology, Model checking.

1 Introduction

There has recently been considerable interest in Semantic Web and rule-based approaches to various aspects of agent technology. The integration of the Semantic Web and intelligent agents research has been realized [11,35], and intelligent agents are considered as a promising approach towards realizing the Semantic Web vision [20]. In order to make better use of the artificial intelligent research experience and the practical application of Semantic Web technologies, the World Wide Web Consortium (W3C) has adopted the Web ontology language (OWL) as a language for processing Web information [8]. While the adoption of the standard ontology language OWL provides the basis for some forms of reasoning, it has been realized that rules are a key element of the Semantic Web vision [22]. For example, ontology based reasoning using DAML/OWL and, more generally, in rule extensions to ontologies (OWL extended with Horn-style rules) such as ORL [22] and SWRL [23] significantly increase the expressive power of ontology languages [22]. While rule-based systems are rapidly becoming an important

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 312–331, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

component of Semantic Web application, the resulting system behavior and the resources required to realize them, namely, how to ensure the correctness of rule-based designs (will a rule-based system produce the correct output for all legal inputs), termination (will a rule-based system produce an output at all) and response time (how much computation will a rule-based system have to do before it generates an output) can be difficult to predict. These problems become even more challenging for distributed rule-based systems, where the system being designed or analyzed consists of several communicating rule-based programs which exchange information via messages. A communicated fact may be added asynchronously to the state of a rule-based system while the system is running, potentially triggering a new strand of computation which executes in parallel with current processing. In order to provide response time guarantees for such systems, we must know how much time a rule-based system needs to perform the required reasoning. Furthermore, for a rule-based system running on a PDA or other mobile device, the number of messages exchanged may be a critical factor.

In this paper, we present a verification framework for an ontology-driven system that supports automated verification of time and communication requirements in distributed rule-based agents. We consider distributed problem-solving in systems of communicating rule-based agents, and ask how much time (measured as the number of rule firings) and how many message exchanges it takes the system to find a solution. We use standard model checking techniques to verify interesting properties of such systems, and show how the Maude model checker can be used to verify properties including response-time guarantees of the form: if the system receives a query, then a response will be produced within n time steps.

The remainder of the paper is organized as follows. In section 2 we provide an overview of ontologies and how rules are derived from ontologies which are the basis of designing our rule-based agents. In section 3 we describe our model of communicating ontology-driven rule-based agents. An example illustrating the approach is provided in section 4. In section 5 we briefly describe a tool Tovrba for translating ontology based specification of the agents into Maude, and in section 6 we present some experimental results using Tovrba. We discuss related work in section 7 and conclude in section 8.

2 Ontology-Driven Horn Clause Rules

Ontologies and rules play a central role in the design and development of Semantic Web applications. In this section, first we briefly discuss ontologies and their integration with rules. Then we show how we extract Horn clause rules from ontologies to design our rule-based agents.

2.1 Ontologies and Rules

An ontology is an explicit formal specification of a conceptualization which defines certain terms of a domain and the relationships among them [18]. The Web

oarieli@mta.ac.il

ontology language OWL is a semantic markup language for ontologies that provides a formal syntax and semantics for them. The W3C declared two different standardizations for OWL: OWL 1 [27] and OWL 2 [28]. The first standardization has three profiles, namely OWL Lite, OWL DL, and OWL Full. The OWL Lite profile is less expressive than the other two profiles and the OWL DL profile is based on description logic (DL). Description logic based OWL is a good candidate for defining ontologies where automated reasoning is required to exploit reasoning algorithms. The OWL Full profile has the maximal expressive power but reasoning over an ontology using all features of OWL Full is undecidable [27]. For efficient and tractable reasoning, a new standardization was declared called OWL 2, which trades some expressive power for more efficient reasoning [26]. OWL 2 DL has three sublanguages known as profiles, namely, OWL 2 EL, OWL 2 QL, and OWL 2 RL; each of them is useful in different application scenarios. All the profiles exhibit a polynomial time complexity for ontological reasoning tasks. For the rest of this paper we refer OWL 2 DL as OWL 2.

In this work, we use the OWL 2 RL and SWRL [23] languages for defining ontologies and rules. OWL 2 RL is suitable for the design and development of rule-based systems. The inspiration behind the design of OWL 2 RL is pD^* [24] and description logic programs (DLP) [17]. OWL 2 RL describes the domain of an ontology in terms of classes, properties, individuals, and datatypes and values. Individual names refer to elements of the domain, e.g., `Mary`; classes describe sets of individuals having similar characteristics, e.g., `Patient`; properties describe binary relationships between pairs of individuals, e.g., `isFeeling` describes a relationship between two individuals `Mary` and `MucositisPainTwo`. Here `Mary` is an individual of the class `Patient` and `MucositisPainTwo` is an individual of the class `Pain`. Notice that properties may be object or data properties depending on the types of individuals. Similarly, datatypes represent types of literal values (e.g., integers). An object property links an individual to an individual, whereas a datatype property links an individual to a data value. In OWL 2 RL, object properties can be functional, inverse functional, reflexive, irreflexive, symmetric, asymmetric, and transitive; however, data properties can be only functional [28]. In addition, complex class descriptions can be constructed using all the above components (classes, properties, individuals, and datatypes and values) and various constructors including union and intersection.

In OWL 2, International Resource Identifiers (IRIs) are used to identify ontologies and their elements. OWL 2 has several syntaxes, including the Manchester Syntax [3] and the Functional-Style Syntax [4]. In this paper, we use the Functional-Style Syntax and we consider the Unique Name Assumption (UNA) for each element of our ontology; the latter allows us to omit the prefix IRI during the translation of an ontology to derive a rule-based system. To illustrate, we present OWL 2 RL axioms (1)-(4) for an ontology about pain management. In order to define the elements of the ontology, we declare a prefix IRI a as `<http://logic.stfx.ca/ontologies/PainOntology.owl#>`. The first axiom defines the concept "Either a doctor or a nurse or a relative of a patient can be a caregiver". The second and third axioms assert the facts that "Mary is a

oarieli@mta.ac.il

patient" and "Mary is feeling Mucositis Pain". The last axiom characterizes the object property `isFeeling` as the inverse object property of `isFeltBy`.

$$Prefix(a :=< http://logic.stfx.ca/ontologies/PainOntology.owl\# >)$$

$$SubClassOf(ObjectUnionOf(DoctorNurseRelative)CareGiver) \quad (1)$$

$$ClassAssertion(a : Patienta : Mary) \quad (2)$$

$$ObjectPropertyAssertion(a : isFeelinga : Marya : MucositisPainTwo) \quad (3)$$

$$InverseObjectProperties(a : isFeelinga : isFeltBy) \quad (4)$$

Both the description logic based OWL 1 and OWL 2 are decidable fragments of first order logic; however, the expressive power of OWL 1 is strictly limited to certain tree structure-like axioms [17]. For instance, a simple rule $hasFather(?x, ?y) \wedge hasBrother(?y, ?z) \rightarrow hasUncle(?x, ?z)$ can not be modeled using OWL 1 axioms. Although OWL 2 can express this *uncle* rule indirectly, many rules are still not possible to model using OWL 2 axioms [25]. Function-free Horn clause rules can remove such restrictions while being decidable but they are restricted to universal quantification and no negation [29]. A combination of OWL 2 with rules offers a more expressive formalism for building Semantic Web applications. Several proposals have been made to combine rules with ontologies. We use one of them, the SWRL that extends OWL DL by adding new axioms, namely Horn clause rules. Although SWRL was a proposed extension for OWL 1, it can be used as an rule extension for OWL 2 [16]. We combine a set of SWRL rules with the set of OWL 2 RL axioms and facts to build our ontology. For example, the statement "if a patient has pain level "2" then the pain intensity (s)he is feeling is mild" can be written in SWRL rule format in Functional-Style syntax as follows:

$$Prefix(var :=< urn : swrl\# >)$$
$$DLSafeRule($$
$$Body($$
$$ClassAtom(a : PatientVariable(var : p))$$
$$ObjectPropertyAtom(a : isFeelingVariable(var : p)Variable(var : x))$$
$$ClassAtom(a : PainVariable(var : x))$$
$$DataPropertyAtom(a : hasPainLevelVariable(var : x)\text{“2”}\hat{}\hat{}xsd : string)$$
$$)$$
$$Head($$
$$ObjectPropertyAtom(a : hasPainIntensityVariable(var : p)a : MildPainIn)$$
$$))$$

More precisely, the SWRL rule defined above states that an individual p from the `Patient` class who is feeling pain x that has pain level "2" asserts the fact that p has pain intensity `MildPainIn`. Here `MildPainIn` is an individual of the class `MildPain`. Such rule is best described in plain text syntax as follows:

$$Patient(?p) \wedge isFeeling(?p, ?x) \wedge Pain(?x) \wedge hasPainLevel(?x, \text{“2”})$$
$$\rightarrow hasPaintIntensity(?p, MildPainIn) \quad (5)$$

oarieli@mta.ac.il

2.2 Translation of Ontologies into Rules

Since OWL 2 RL is based on DLP, the set of axioms and facts of an OWL 2 RL ontology can be translated to Horn clause rules [17]. In order to design an ontology-driven rule-based system, first we use the DLP framework [17] to translate an ontology to a set of Horn clause rules. In OWL 2 RL, facts are described using ClassAssertion and ObjectPropertyAssertion/DataPropertyAssertion which correspond to DL axioms of the form $a : C$ and $\langle a, b\rangle : P$, respectively, where a and b are individuals, C is a class, and P is an object/data property. Note that these facts are already in the Horn clause rule format with empty bodies. For instance, the facts in (2) and (3) are translated as `Patient(Mary)` and `isFeeling(Mary,MucositisPainTwo)`.

Table 1. Translation of OWL 2 RL axioms and facts into Horn clause rules

OWL 2 Axioms and Facts	DL Syntax	Horn clause rule
`ClassAssertions`	a:C	C(a)
`PropertyAssertion`	$\langle a, b\rangle : P$	$P(a, b)$
`SubClassOf`	$C \sqsubseteq D$	$C(x) \rightarrow D(x)$
`EquivalentClasses`	$C \equiv D$	$C(x) \rightarrow D(x)$,$D(x) \leftarrow C(x)$
`EquivalentProperties`	$P \equiv Q$	$Q(x, y) \rightarrow P(x, y)$
		$P(x, y) \rightarrow Q(x, y)$
`ObjectInverseOf`	$P \equiv Q^-$	$P(x, y) \rightarrow Q(y, x)$
		$Q(y, x) \rightarrow P(x, y)$
`TransitiveObjectProperty`	$P^+ \sqsubseteq P$	$P(x, y) \wedge P(y, z) \rightarrow P(x, z)$
`SymmetricObjectProperty`	$P \equiv P^-$	$P(x, y) \rightarrow P(y, x)$
`Object/DataUnionOf`	$C_1 \sqcup C_2 \sqsubseteq D$	$C_1(x) \rightarrow D(x)$, $C_2(x) \rightarrow D(x)$
`Object/DataIntersectionOf`	$C \sqsubseteq D_1 \sqcap D_2$	$C(x) \rightarrow D_1(x)$,$C(x) \rightarrow D_2(x)$
`Object/DataSomeValuesFrom`	$\exists P.C \sqsubseteq D$	$P(x, y) \wedge C(y) \rightarrow D(x)$
`Object/DataAllValuesFrom`	$C \sqsubseteq \forall P.D$	$C(x) \wedge P(x, y) \rightarrow D(y)$
`Object/DataPropertyDomain`	$\top \sqsubseteq \forall P^-.C$	$P(y, x) \rightarrow C(y)$
`Object/DataPropertyRange`	$\top \sqsubseteq \forall P.C$	$P(x, y) \rightarrow C(y)$

The syntax of OWL 2 RL is asymmetric, i.e., the syntactic restrictions allowed for subclass expressions differ from those allowed for superclass expressions. For instance, an existential quantification to a class expression (`ObjectSo- meValuesFrom`) is allowed only in subclass expressions whereas universal quantification to a class expression (`ObjectAllValuesFrom`) is allowed only in superclass expressions. These restrictions facilitate the translation of OWL 2 RL axioms into Horn clause rules based on the DLP framework. Translations of some of the OWL 2 RL axioms and facts into rules are given in Table 1. In the second column, complete DL statements are given which are constructed by the corresponding OWL 2 RL axioms and facts to illustrate the translation. For example, `ObjectIntersectionOf`($\sqcap$) is represented by the statement $C_1 \sqsubseteq D_1 \sqcap D_2$. The translation of SWRL rules is straightforward because they

oarieli@mta.ac.il

are already in the Horn clause rule format. In our approach, agents in a multi-agent rule-based system are designed using the translated Horn clause rules of an ontology. The translation process is automated and is a part of the TOVRBA tool (cf. § 5).

3 Systems of Communicating Rule-Based Agents

We adopt the model of distributed agents presented in [6]. A distributed reasoning system consists of n_{Ag} (≥ 1) individual reasoners or *agents*. Each agent is identified by a value in $\{1, 2, \ldots, n_{Ag}\}$ and we use variables i and j over $\{1, 2, \ldots, n_{Ag}\}$ to refer to agents. An agent in the system is either concrete or abstract. Each concrete agent has a program, consisting of Horn clause rules (derived from OWL 2 RL + SWRL), and a working memory, which contains facts (ground atomic formulae) representing the initial state of the system. The introduction of ontology-driven rules increases the expressiveness of the framework in [6], and makes it easier to model complex real world Semantic Web problems. In addition, existing tools, including Protégé [5], support the design of OWL 2 RL + SWRL based ontologies, making it easier to model rule-based agents using semantic rules. The behavior of each abstract agent is represented in terms of a set of temporal doxastic formulae. A doxastic logic is a modal logic concerned with reasoning about beliefs. That is, abstract specifications are given as LTL formulae which describe the external behavior of agents, and allow their temporal behavior (the response time behavior of the agent), to be compactly modeled. The agents (concrete and abstract) execute synchronously. We assume that each agent executes in a separate process and that agents communicate via message passing. We further assume that each agent can communicate with multiple agents in the system at the same time.

3.1 Concrete Agents

The two main components of rule-based agents are the knowledge base (KB) which contains a set of IF-THEN rules and the working memory (WM) which contains a set of facts that constitute the current (local) state of the system. Another component of a rule-based system is the inference engine which reasons over rules when the application is executed. The inference engine may have some reasoning strategies to handle cases when multiple rule instances are eligible to fire.

In Listing 1.1, we specify the abstract syntax for concrete agents rules using a BNF. In this notation, the terminals are quoted, the non-terminals are not quoted, alternatives are separated by vertical bars, and components that can occur zero or more times are enclosed braces followed by a superscript asterisk symbol ($\{\ldots\}^*$). A class atom represented by `description(i-object)` in the BNF consists of an OWL 2-named class and a single argument representing an OWL 2 individual, for example an atom `Person(x)` holds if `x` is an instance of the class description `Person`. Similarly, an individual property atom represented by `individualvaluedProperty(i-object,i-object)` consists of an OWL 2

oarieli@mta.ac.il

object property and two arguments representing OWL 2 individuals, for example an atom `hasCarer(x,y)` holds if `x` is related to `y` by property `hasCarer` and so on. Note that OWL 2 is limited to unary and binary predicates and it is function-free. Therefore, in the Protégé editor all the arguments of *Ask* and *Tell* are represented using constant symbols and these annotated symbols are translated appropriately when designing the target system using the Maude specification.

```
Rule ::= '<' Priority ':' Atoms '→' Atom '>'
Atoms ::= Atom {∧ Atom}*
Atom ::= standardAtom | commmunicationAtom
standardAtom ::= description'('i-object ')'
            | individualvaluedProperty'('i-object ',' i-object ')'
            | datavaluedProperty'('i-object ',' d-object ')'
            | sameIndividuals'('i-object ',' i-object ')'
            | differentIndividuals'('i-object ',' i-object ')'
            | dataRange'(' d-object ')'
            | builtIn'(' builtinId ',' {d-object}* ')'
communicationAtom ::= 'Ask(' i ',' j ',' standardAtom ')'
            | 'Tell(' i ',' j ',' standardAtom ')'
Priority ::= N≥0
N≥0 ::= 0 | 1 | 2 | ...
i ::= 1 | 2 | ... | nAg
j ::= 1 | 2 | ... | nAg
builtinID ::= URIreference
i-object ::= i-variable | individualID
d-object ::= d-variable | dataLiteral
i-variable ::= 'I-variable('URIreference')'
d-variable ::= 'D-variable('URIreference')'
```

Listing 1.1. Abstract syntax for concrete agents rules

Rules derived from OWL 2 RL + SWRL are translated using Tovrba into the simplified text format for the encoding and verification of the target system. In other words, the rules of a concrete agent have the plain text format $\langle\ n\ :\ P_1 \wedge P_2 \wedge \ldots \wedge P_n \rightarrow P\ \rangle$ where n is a constant that represents the annotated priority of the rule and the P_i's and P are atoms. For communication, we assume a simple query-response scheme based on asynchronous message passing. Each agent's rules may contain two distinguished communication atoms: $Ask(i, j, P)$, and $Tell(i, j, P)$, where i and j are agents and P is an atomic formula not containing an *Ask* or a *Tell*. $Ask(i, j, P)$ means 'i asks j whether P is the case' and $Tell(i, j, P)$ means 'i tells j that P' ($i \neq j$). The positions in which the *Ask* and *Tell* primitives may appear in a rule depends on which agent's program the rule belongs to. Agent i may have an *Ask* or a *Tell* with arguments (i, j, P) in the consequent of a rule; for example, $\langle\ n\ :\ P_1 \wedge P_2 \wedge \ldots \wedge P_n \rightarrow\ Ask(i, j, P)\ \rangle$ whereas agent j may have an *Ask* or a *Tell* with arguments (i, j, P) in the antecedent of the rule; for example, $\langle\ n\ :\ Tell(i, j, P) \rightarrow P\ \rangle$ is a well-formed rule (we call it trust rule) for agent j that causes it to believe i when i informs it that P is the case. No other occurrences of *Ask* or *Tell* are allowed. When a rule has either an *Ask* or a *Tell* as its consequent, we call it a communication rule. All other rules are known as deduction rules. These include rules with *Ask*s and *Tell*s in the antecedent as well as rules containing neither an *Ask* nor a *Tell*.

oarieli@mta.ac.il

Firing a communication rule instance with the consequent $Ask(i, j, P)$ adds the atom $Ask(i, j, P)$ both to the working memory of i and of j. Intuitively, i has a record that it asked j whether P is the case, and j has a record of being asked by i whether P is the case. Similarly, if the consequent of a communication rule instance is of the form $Tell(i, j, P)$, then the corresponding atom $Tell(i, j, P)$ is added to the working memories of both the agents i and j. The set of facts are ground atomic formulae.

We assume that each concrete agent has a *reasoning strategy* (or conflict resolution strategy) which determines the order in which rules are applied when more than one rule matches the contents of the agent's working memory. The framework (and the TOVRBA tool presented in §5) supports a set of standard conflict resolution strategies often used in rule-based systems [12,15,37] including: rule ordering, depth, breadth, simplicity, and complexity. Different agents in the system may use different types of reasoning strategies. To allow the implementation of reasoning strategies, each atom is associated with a time stamp which records the cycle at which the atom was added to the working memory. The internal configurations of the rules follow the syntax given below:

$$\langle\ n\ :\ [\ t_1\ :\ P_1] \wedge [\ t_2\ :\ P_2] \wedge \ldots \wedge [\ t_n\ :\ P_n] \rightarrow [\ t\ :\ P]\ \rangle$$

where the t_i's and t represent time stamps of atoms. When a rule instance of the above rule is fired, its consequent atom P will be added to the working memory with time stamp $t = t' + 1$, i.e., t will be replaced by $t' + 1$, where t' is the current cycle time of the system.

3.2 Abstract Agents

An abstract agent consists of a working memory and a behavioral specification. The behavior of abstract agents is specified using the temporal logic LTL extended with belief operators. The decision regarding which agents to abstract and how their external behavior should be specified rests with the system designer. Specifications of the external (observable) behavior of abstract agents may be derived from, e.g., assumed characteristics of as-yet-unimplemented parts of the system, or from assumptions regarding the behavior of parts of the overall system the designer does not control (e.g., quality of service guarantees offered by an existing web service) or from the prior verification of the behavior of other (concrete) agents in the system. The general form of the formulae used to represent the external behavior of an abstract agent i is given below:

$$\begin{array}{lll}
\rho & ::= & X^{\leq n}\varphi_1 \mid G(\varphi_2 \rightarrow X^{\leq n}\ \varphi_3) \\
\varphi_1 & ::= & B_i\ Ask(i, j, P) \mid B_i\ Tell(i, j, P) \\
 & & \mid B_i\ Ask(j, i, P) \mid B_i\ Tell(j, i, P) \\
 & & \mid B_i\ P \\
\varphi_2 & ::= & B_i\ Ask(j, i, P) \mid B_i\ Tell(j, i, P) \\
\varphi_3 & ::= & B_i\ Tell(i, j, P) \mid B_i\ Tell(i, k, P) \\
 & & \mid B_i\ Ask(i, j, P) \mid B_i\ Ask(i, k, P)
\end{array}$$

where X is the next step temporal operator, $X^{\leq n}$ is a sequence of n X operators, G is the temporal 'in all future states' operator, and B_i for each agent i is a

oarieli@mta.ac.il

syntactic doxastic operator used to specify agent i's 'beliefs', i.e., the contents of its working memory. Formulae of the form $X^{\leq n}\varphi_1$ describe agents which produce a certain message or input to the system within n time steps. A formula φ_1 of the form $B_i\ Ask(i,j,P)$ or $B_i\ Tell(i,j,P)$ results in communication with the other agent as follows: when the beliefs appear (as an Ask or a $Tell$) in the abstract agent i's working memory, they are also copied to agent j's working memory at the next step. A formula of the form $B_i\ P$ representing a belief involving an atom P (other than Ask and $Tell$) may also appear in the abstract agent i's working memory within n time steps. This is not critical to how abstract agents interact with communication; however it describes agent i's own behavior.

The $G(\varphi_2 \rightarrow X^{\leq n}\ \varphi_3)$ formulae describe agents which are always guaranteed to reply to a request for information within n time steps. We interpret the formula $G(B_i\ Ask(j,i,P) \rightarrow X^{\leq n}\ B_i\ Tell(i,j,P))$ as follows: if t is the time stamp when abstract agent i came to believe formula $Ask(j,i,P)$ (agent j asked for P), then the formula $Tell(i,j,P)$ must appear in the working memory of agent i within $t+n$ steps. The formula $Tell(i,j,P)$ is then copied to agent j's working memory at the next step. The other possible combinations of Ask and $Tell$ in places of φ_2 and φ_3 in the $G(\varphi_2 \rightarrow X^{\leq n}\ \varphi_3)$ formulae can be interpreted in a similar way. Note that we do not need the full language of LTL (for example, the Until operator) in order to specify these abstract agents, and the language described above for abstract agents is independent of the language of concrete agents.

4 Example of Two Communicating Agents

To illustrate the use of the proposed framework, let us consider an example system consisting of two agents where one is concrete and the other is abstract. We consider a scenario in which agent 1, the concrete agent, has the following set of rules:

$$\begin{aligned}
&Rule1 < 1 : prescribedOpioid(?p,' NonOpioidRegimen) \\
&\qquad \rightarrow Ask(1, 2, needAssessment(?p)) > \\
&Rule2 < 2 : Tell(2, 1, isFeeling(?p, ?x)) \rightarrow isFeeling(?p, ?x) > \\
&Rule3 < 3 : Patient(?p) \wedge isFeeling(?p, ?x) \wedge Pain(?x) \wedge hasPainLevel(?x,' 2) \\
&\qquad \rightarrow hasPaintIntensity(?p,' MildPainIn) \\
&Rule4 < 4 : hasPaintIntensity(?p,' MildPainIn) \\
&\qquad \rightarrow prescribedOpioid(?p,' WeakOpioidRegimen) >
\end{aligned}$$

For the rest of this paper constants are preceded by a single quote (note that ' and ′ have the same meaning). The first rule states that if a patient is currently in the non-opioid regimen then ask the abstract agent 2 for an assessment. The second rule is a trust rule for agent 1 which makes it trust agent 2 when agent 2 informs it, say for example, `isFeeling('Mary,'MucositisPainTwo)` which is a ground instantiation of `isFeeling(?p, ?x)`. The third rule states that if a patient has a pain level "2" then the pain intensity (s)he is feeling is mild. The fourth rule state that if a patient feels mild pain then the prescribed opioid

oarieli@mta.ac.il

regimen is weak. The external behavior of the abstract agent 2 is described by the following temporal logic formula:

Table 2. Example derivation of two communicating agents

Time	Agent 1	Agent 2
0	{[0 : Pain('MucositisPainTwo)] [0 : Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)]}	{ }
	Rule1	Idle
1	{[0 : Pain('MucositisPainTwo)] [0 : Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)] [1 : Ask(1,2,needAssessment('Mary))]}	{ }
	Idle	Copy (Ask(1,2,needAssessment('Mary)) from Agent 1)
2	{[0 : Pain('MucositisPainTwo)] [0 : Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)] [1 : Ask(1,2,needAssessment('Mary))]}	{[2 : Ask(1,2,needAssessment('Mary))] }
	Idle	Idle
3	{[0 : Pain('MucositisPainTwo)] [0:Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)] [1 : Ask(1,2,needAssessment('Mary))]}	{[2 : Ask(1,2,needAssessment('Mary))] }
	Idle	Idle
4	{[0 : Pain('MucositisPainTwo)] [0 : Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)] [1 : Ask(1,2,needAssessment('Mary))]}	{[2 : Ask(1,2,needAssessment('Mary))] }
	Idle	$Tell$
5	{[0 : Pain('MucositisPainTwo)] [0 : Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)] [1 : Ask(1,2,needAssessment('Mary))]}	{[2 : Ask(1,2,needAssessment('Mary))] } [5 : $Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo))] }
	Copy ($Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo)) from Agent 2)	Idle
6	{[0 : Pain('MucositisPainTwo)] [0 : Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)] [1 : Ask(1,2,needAssessment('Mary))] [6 : $Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo))]}	{[2 : Ask(1,2,needAssessment('Mary))] } [5 : $Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo))] }
	Rule2	Idle
7	{[0 : Pain('MucositisPainTwo)] [0 : Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)] [1 : Ask(1,2,needAssessment('Mary))] [6 : $Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo))] [7 : isFeeling('Mary, 'MucositisPainTwo)]}	{[2 : Ask(1,2,needAssessment('Mary))] } [5 : $Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo))] }
	Rule3	Idle
8	{[0 : Pain('MucositisPainTwo)] [0 : Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)] [1 : Ask(1,2,needAssessment('Mary))] [6 : $Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo))] [7 : isFeeling('Mary, 'MucositisPainTwo)] [8 : hasPaintIntensity('Mary,'MildPainIn)]}	{[2 : Ask(1,2,needAssessment('Mary))] } [5 : $Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo))] }
	Rule4	Idle
9	{[0 : Pain('MucositisPainTwo)] [0 : Patient('Mary)] [0 : prescribedOpioid('Mary, 'NonOpioidRegimen)] [0 : hasPainLevel('MucositisPainTwo,'2)] [1 : Ask(1,2,needAssessment('Mary))] [6 : $Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo))] [7 : isFeeling('Mary, 'MucositisPainTwo)] [8 : hasPaintIntensity('Mary,'MildPainIn)] [9 : prescribedOpioid('Mary, 'WeakOpioidRegimen)]}	{[2 : Ask(1,2,needAssessment('Mary))] } [5 : $Tell$(2,1,isFeeling('Mary, 'MucositisPainTwo))] }

oarieli@mta.ac.il

$G(B_2 \;\; Ask(1, 2, needAssessment('Mary)) \rightarrow X^{\leq 3} B_2 \;\; Tell(2, 1, isFeeling('Mary,' MucositisPainTwo)))$

which states that whenever the atom Ask`(1,2,needAssessment('Mary))` appears in the working memory of agent 2 (i.e., agent 2 believes the atom Ask`(1,2, needAssessment('Mary))`), the atom $Tell$`(2,1,isFeeling('Mary,'MucositisPainTwo))` will appear in the working memory of agent 2 (i.e., agent 2 will believe atom $Tell$`(2,1,isFeeling('Mary,'MucositisPainTwo))`) within 3 time steps. Suppose now that the initial working memories of the agents as follows:

WM_1 :{ [0: prescribedOpioid('Mary,'NonOpioidRegimen)] [0 : Patient('Mary)]
[0: hasPainLevel('MucositisPainTwo,'2)] [0 : Pain('MucositisPainTwo)]}
WM_2 :{ }.

Table 2 gives a simple example of a run of the system starting from the initial configuration. This example helps to explain how facts are derived and communicated, and what happens when the abstract agent receives an Ask query by communication. Note that in this derivation it is assumed that at step 2, when abstract agent 2 came to believe atom Ask`(1,2,needAssessment('Mary))` (i.e., agent 1 asked for `needAssessment('Mary)`), the atom $Tell$`(2,1,isFeeling('Mary,'MucositisPainTwo))` appeared in the working memory of agent 2 at time $2 + 3$ i.e., at the 5th step. However, the atom $Tell$`(2,1,isFeeling('Mary, 'MucositisPainTwo))` could also appear at the 3rd or 4th step but definitely appears at step 5 if it is not already present in the working memory of agent 2. In the example run, at the 2nd and 3rd steps both agents perform an Idle action. In this model, communication requires a single time step, i.e., when agent i asks a query (or tells an information to) agent j at time step t, agent j will receive the query (information) at time step $t+1$. The table shows that from the initial configuration agents can derive `prescribedOpioid('Mary, 'WeakOpioidRegimen)` (appears in the configuration of one of the agents) in 9 time steps and exchange two messages.

5 Automated Verification Tool Tovrba

We use the Protégé version 4.1 ontology editor and knowledge-base framework to build the ontologies and define the rules for concrete agents. The SWRL editor is integrated with Protégé and permits the interactive editing of SWRL rules. In order to encode an ontology-driven rule-based system using a Maude [10] specification and formally verify its interesting properties, we first need to translate the ontology in the OWL/XML format to a set of simple plain text Horn clause rules. We developed a translator that takes as input an OWL 2 RL ontology in the OWL/XML format (an output file of the Protégé editor) and translates it to a set of plain text Horn clause rules. We use the OWL API [21] to parse the ontology and extract the set of axioms and facts. The design of the OWL API is directly based on the OWL 2 Structural Specification and it treats an ontology as a set of axioms and facts which are read using the visitor design pattern. The DLP-based translation rules (cf. § 2.2) are then recursively applied

oarieli@mta.ac.il

to generate equivalent plain text Horn clause rules for each axiom and fact. We also extract the set of SWRL rules using the OWL API which are already in the Horn clause rule format. First, atoms with corresponding arguments associated with the head and the body of a rule are identified and we then generate a plain text Horn clause rule for each SWRL rule using these atoms.

The translated Horn clause rules of an ontology are then used to create agents of a multi-agent rule-based system using the Maude specification. We then automatically verify interesting properties of the system using the Maude LTL model checker. The high-level architecture of the TOVRBA tool is shown in Figure 1. We use the Maude rewriting system because it allows efficient modeling of the agents' first order rules and reasoning strategies. For example, the variables that appear in a rule can be represented directly in the Maude encoding, without having to generate all ground instances resulting from possible variable substitutions.

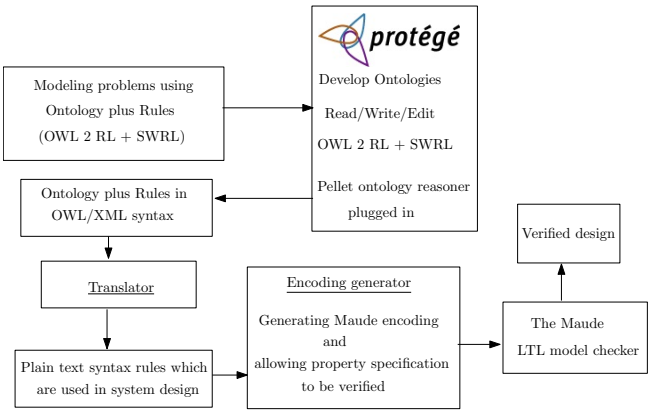


Fig. 1. The TOVRBA tool architecture

In [7] a preliminary description of the tool was published for rule-based multi-agent systems which allows the designer to specify information about agents interactions, behavior, and execution strategies at different levels of abstraction. The extended version of the tool presented in this paper allows the designer to model ontology-driven rule-based agents consisting of rules derived from OWL 2 RL + SWRL. Each agent in a multi-agent system has a configuration (local state) and the composition of all these configurations (local states) make the configuration (global state) of the multi-agent system. The types necessary to implement the local state of an agent (working memory, program, agenda, reasoning strategies, message counters, time stamps etc.) are declared in a generic agent configuration functional module called `ACM`. The structure of the `ACM` is given in Listing 1.2.

oarieli@mta.ac.il

```
fmod ACM is
  protecting NAT .
  protecting BOOL .
  protecting QID .
  sorts Constant Atom sAtom cAtom Term Rule Agenda WM .
  sorts TimeA TimeWM RepT RepTime Config .
  subsort Atom < WM .
  subsorts aAtom cAtom < Atom .
  subsort Rule < Agenda .
  subsort Qid < Constant .
  subsort TimeA < TimeWM .
  subsorts Constant < Term .
  subsort RepT < RepTime .
  ops void rule : -> Atom .
  ops com exec : -> Phase [ctor] .
  op nil : -> Term[ctor] .
  op [_ : _] : Nat Atom -> TimeA .
  op _ _ : WM WM -> WM [comm assoc] .
  op _ _ : TimeWM TimeWM -> TimeWM [comm assoc] .
  op _ _ : Agenda Agenda -> Agenda [comm assoc] .
  op <_ : _->_> : Nat TimeWM TimeA -> Rule .
  op _ _ : RepTime RepTime -> RepTime [comm assoc] .
  op Ask : Nat Nat sAtom -> cAtom .
  op Tell : Nat Nat sAtom -> cAtom .
    .
    .
    .
endfm
```

Listing 1.2. Sorts declaration and their relationships

A number of Maude library modules such as `NAT`, `BOOL`, and `QID` have been imported into the `ACM` functional module. The modules `NAT` and `BOOL` are used to define natural and Boolean values, respectively, whereas the module `QID` is used to define the set of constant symbols (constant terms of the rule-based system). The set of variable symbols (variable terms of the rule-based system) are simply Maude variables of sort `QID`. Both variables and constants are subsorts of sort `Term`. Similarly, an atom is declared as an operator whose arguments are of sort `Term`, and returns an element of sort `Atom`. For example, `op isFeeling : Term Term -> Atom`. The sort `Atom` is declared as a subsort of the sort `WM` (working memory), and a concatenation operator is declared on sort `WM` which is the double underscore shown below.

```
op _ _ : WM WM -> WM [comm assoc] .
```

The above operation is in mixfix notation and it is commutative and associative. This means that working memory elements are a set of atoms whose order does not matter. In order to maintain time stamps for each atom, a sort `TimeA` is declared whose elements are of the form `[ t : P ]`, where `t` represents the time stamp of atom `P` indicating when that atom was added to the working memory. The sort `TimeA` is declared as a subsort of the sort `TimeWM`, and a concatenation operator is declared on sort `TimeWM` which is also the double underscore and is commutative and associative.

```
op _ _ : TimeWM TimeWM -> TimeWM [comm assoc] .
```

Note that `WM` and `TimeWM` are updated simultaneously, for example, whenever an atom `P` is added to `WM` the corresponding element `[ t : P ]` is also added

oarieli@mta.ac.il

to `TimeWM` for an appropriate time step `t`. Fact time stamps are maintained to implement reasoning strategies. The rules of each agent are defined using an operator which takes as arguments a sort `Nat` specifying the priority, a set of atoms (of sort `TimeWM`) specifying the antecedents of the rule and a single atom (of sort `TimeA`) specifying the consequent, and returns an element of sort `Rule` (line 21 of Listing 1.2). The sort `Rule` is declared as a subsort of the sort `Agenda`, and a concatenation operator is declared on sort `Agenda` which is also the double underscore and is commutative and associative.

```
op _ _ : Agenda Agenda -> Agenda [comm assoc] .
```

In a similar fashion, other sorts and operators are declared and manipulated. We model each (concrete and abstract) agent using a functional module which imports the `ACM` module defined above. The local configuration of an agent i is represented as a tuple `Si[A|RL|TM|M|RT|RT'|t|msg|syn]iS`, where `Si` and `iS` indicate start and end of a state of agent i. The variables `A` and `RL` are of sort `Agenda`, `TM` is of sort `TimeWM`, `M` is of sort `WM`, `RT` and `RT'` are of sort `RepTime`. Moreover, `t`, `msg`, and `syn` are of sort `Nat`. The variables `t`, `msg`, and `syn` have been used to represent respectively the time step, message counter, and a flag for synchronization. Note that the structure of local configurations for both concrete and abstract agents are the same. This is to maintain consistency of the shape of each agent's configuration. However, for example, the sort `RepTime` is of no use for concrete agents and its value is always empty for them. For the sake of brevity, we do not describe the encoding in any further details here, we refer the interested reader to [32].

6 Pain Management System

In this section, we consider an example system adopted from the guidelines for the management of cancer related pain in adults [9]. A simplified flowchart of the guidelines is shown in Figure 2. The guidelines say that if a patient is not currently on a regular opioid regimen then depending on the pain intensity of a patient (s)he will be placed on one of the three regimens non-opioid, weak opioid, or strong opioid. Depending on the patient's response the opioid may be changed, e.g., from non-opioid to weak opioid or weak opioid to strong opioid. The guidelines for the strong opioid regimen say that if a patient is responding (i.e., if the current pain level is equal or less than previous pain level) then another reassessment should be done at an appropriate interval, say within a week. However, if a patient is not responding then it suggests a different reassessment interval depending on the current pain level and considers increasing the dose by a certain percentage. The reassement also suggests that if there are side effects (side effects are symptoms which may occur at any dose e.g., constipation) then management of side effects from opioid therapy will take effect. Also, when on the strong opioid regimen, if a pain crisis occurs with any patient at any point in time, it is an emergency situation.

We built a pain monitoring ontology, that integrates the terminology and concepts of health and medicine used in the Guysborough Antigonish Strait Health

oarieli@mta.ac.il

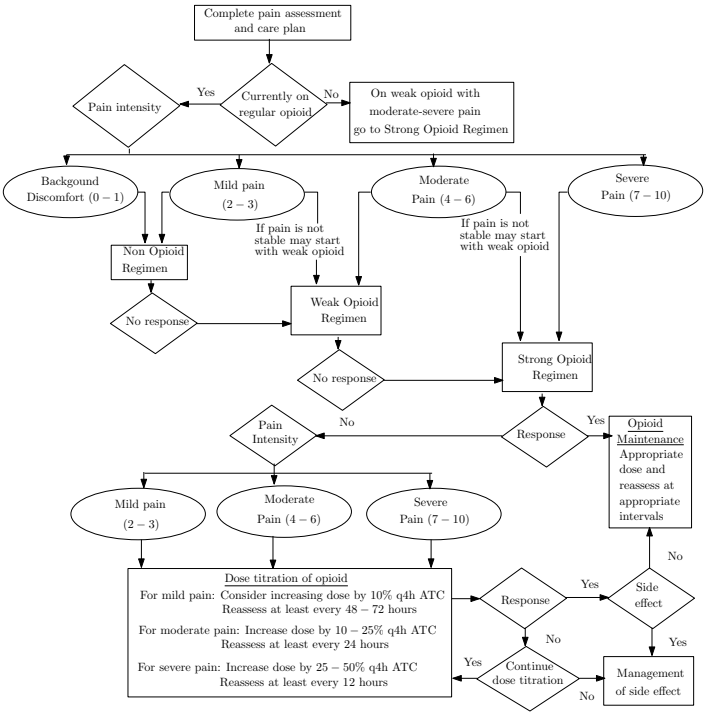


Fig. 2. Cancer pain treatment guidelines [9]

Authority (GASHA). A set of standard terms were also obtained from SNOMED-CT [2], ICNP [19], and the guidelines for Cancer pain treatment. We followed the Basic Formal Ontology [1] which is a philosophically inspired top level ontology that provides a coherent, unified understanding of the whole domain. A fragment of the pain monitoring ontology is depicted in Figure 3. In our ontology, we have several classes including `Pain`, `Person`, `Patient`, `PainintensityType`, `SpecialPainProblem`, `SideEffects`, object properties include, e.g., `hasPainIntensity, Domain:Pain, Range:PainIntensityType`, and data properties include, e.g., `hasPainLevel, Domain:Pain, Range:xsd:int`. It also includes inverse object properties such as, for example, `isFeeling` and `isFeltBy` and functional object properties, e.g., `hasPainLevel`, i.e., each pain level belongs to an instance of `Pain` class. We also use propositional connectives to create complex class expressions; an example of such expressions is given in axiom 1 (cf. § 2.1).

We model a multi-agent rule-based system using Tovrba based on the above defined pain monitoring ontology. The system consists of one concrete agent and several abstract agents. The concrete agent in the system is modeled as a central Health Planner *planner*(p). We model five abstract agents, namely, an assessor *assessor* (a), a reassessor *reassessor*(r), a side effect manager *sideeffectmanager* (s), a care giver *caregiver*(c), and an emergency *emergency*(e). These abstract agents interact with the agent p and they communicate via message passing.

oarieli@mta.ac.il

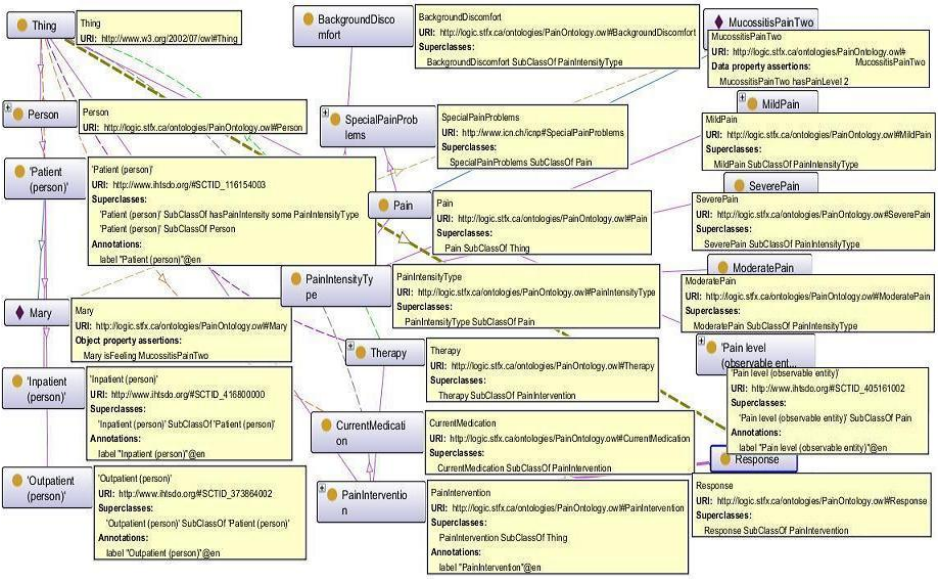


Fig. 3. A fragment of the pain monitoring ontology

For simplicity, we assume that agent a can be a patient or a family member, agent r can be a physician or a nurse or a relative or a patient herself, and agent c can be a physician or a nurse.

Example rules for the concrete agent p are:

$$< 2 : Patient(?p) \wedge isFeeling(?p, ?x) \wedge Pain(?x) \wedge hasPainLevel(?x,' 2) \\ \rightarrow hasPaintIntensity(?p,' MildPainIn) >$$

$$< 2 : Patient(?p) \wedge isFeeling(?p, ?x) \wedge Pain(?x) \wedge hasPainLevel(?x,' 1) \\ \rightarrow hasPaintIntensity(?p,' BackgroundDiscomfortIn) >$$

The abstract agent a assesses the current pain level of a patient and informs the planner p, with messages of the form: $Tell(a, p, isFeeling(?p, ?x))$.

Upon receiving the pain level information from agent a, agent p derives (prescribes) a medication for the patient by firing a sequence of rules from its knowledge base. The type of medication could be non-opioid, weak-opioid, or strong-opioid depending on the levels of pain. Agent p then asks agent a if the patient is responding. Depending on the response, agent p suggests either to continue the current medication or change from the current opioid to another opioid. If a patient is on the strong opioid regimen, when agent p interacts with agent r for the reassessment, agent p also asks r if there is any side effect during medication. If at any point in time agent r informs agent p about a pain crisis, then the planner will contact agent c. If agent p receives a negative (notAvailable or Busy) response from agent c then planner p contacts agent e. Agent a in the system generates information about the pain levels at different times in the interval $[1, 5]$.

oarieli@mta.ac.il

For example, agent a generates pain information for a patient named `Mary` who is feeling `MucositisPainOne` using the following formula:

$$X^5 \;\; B_a \;\; Tell(a, p, isFeeling('Mary,' MucositisPainOne))$$

Similarly, the behavior of agent a is also represented, for example, using formulas of the form:

$$G(B_a \;\; Ask(p, a, needAssessment('Mary)) \rightarrow X^{\leq 4} B_a \;\; Tell(a, p, isFeeling('Mary,' MucositisPainSeven)))$$

We verify the following properties of the system:

$$G(\;\; B_p \;\; Tell(a, p, isFeeling('Mary,' MucositisPainOne)) \rightarrow X^n \;\; B_p \;\; hasPaintIntensity('Mary,' BackgroundDiscomfortIn) \;\;)$$

the above property specifies that whenever agent a tells agent p that `Mary` is feeling `MucositisPainOne`, agent p classifies `Mary`'s pain intensity as Background Discomfort within n time steps and

$$G(\;\; B_p \;\; hasCarer('Mary,' John) \wedge B_p \;\; Tell(c, p, hasAcknowledgement('John,' Busy)) \wedge B_p \;\; hasPainCrisis('Mary) \rightarrow X^n \;\; B_p \;\; Tell(p, e, hasPainCrisis('Mary)) \;\;)$$

which specifies that whenever a pain crisis occurs with a patient and agent p has received negative acknowledgment from agent c, agent p contacts the emergency agent e within n time steps.

The above properties are verified as true when the value of n is 2 in the first property, and 3 in the second property; and the model checker uses 2 seconds for each property. However, when we assign a value to n which is less than 2 in the first property, and less than 3 in the second property, the properties are verified as false and the model checker returns counterexamples. This also demonstrates the correctness of the encoding in that the model checker does not return true for arbitrary values of n. Note that in our experiment, we use discrete time steps; however, we can map the value of n into minutes, for example n equals 3 is treated as sixty minutes.

7 Related Work

There has been considerable work on the Semantic Web and rule-based agents, both in AI and in the active database community. In [35], Subercaze and Maret present a semantic agent model that allows SWRL programming of agents. A Java interpreter has been developed that communicates with the Knowledge Base using the Protégé-OWL API. The prototype tool takes advantages of the Java-based domain modeling tool JADE that allows agent registration, service discovery and messages passing. The framework supports FIPA-ACL for agent communication.

In [30], Mousavi et al. present an ontology-driven reasoning system based on BDI agent model [33]. In contrast to Jadex (that utilizes an XML format to represent agents' plans, beliefs and goals), in their framework, an ontology (in an OWL format) has been used to represent agents' believes, plans and events.

oarieli@mta.ac.il

The Java-based tool JADE was used to implement the agents, and the Protégé OWL was used to create the ontology. To illustrate the use of the framework, a simple Mobile Workforce Brokering Systems (a multi-agent system that automates the process of allocating tasks to Mobile Workforces) was modeled for simulation.

In [34] Ruan and MacCaull present an approach to monitor healthcare workflows using a logic-based formal method. To specify the system the authors have presented FO-LTL-K a logic fusion of first order LTL and description logic. They have shown how some of the norms in [14] can be specified using FO-LTL-K, however, the paper produced no practical results. In [31] Rabbi et al. present a different approach for modeling and verifying compensable healthcare workflow with time constraints and monitors. They use NOVA workflow tool for the modeling and automated translation of the monitoring system into the specification language of DiVinE model checker to verify properties of the system. However their framework is neither ontology nor agent based.

In [13], Dekhtyar et al. studied the complexity of verification of behavior (dynamic) properties of deterministic, nondeterministic and asynchronous multi-agent systems. They considered the MAS framework based on the IMPACT architecture presented in [36]. In order to analyze complexities in MAS verification, the authors impose easy-to-formulate limitations on IMPACT agents, which lead to a polynomial time semantics of the systems. Although in this paper we do not intend to analyze the complexity of MAS verification, the work presented in [13] motivates us to explore some research in this direction.

While in the above a number of ontology-driven modeling and reasoning approaches [30,35] have been developed for multi-agent systems, to our knowledge tools for automated formal verification for such systems are lacking.

8 Conclusions

In this paper, we proposed an approach to modeling and verifying response time guarantees of ontology-driven multi-agent rule-based systems. We use standard model checking techniques to verify interesting properties of such systems, and show how the Maude LTL model checker can be used to verify properties including response-time guarantees of the form: if the system receives a query, then a response will be produced within n time steps. We described results of experiments on a simple healthcare monitoring system. In future work, we plan to evaluate our approach on more real-life examples of Semantic Web and rule-based systems, and enhance our framework with context-aware capabilities.

Acknowledgments. This work is supported by an ACEnet Post Doctoral Fellowship, an NSERC Industrial Post Graduate Fellowship, and ACOA. The computational facilities are provided by ACEnet. We would like to thank Rachel Embree and Mary Heather Jewers for the fruitful discussions about ontologies and the guidelines for the management of cancer related pain in adults [9].

oarieli@mta.ac.il

References

1. Basic formal ontology (2002), http://ontology.buffalo.edu/bfo/
2. SNOMED-CT Systematized Nomenclature of Medicine-Clinical Terms (2007), http://www.ihtsdo.org/snomed-ct/
3. OWL 2 Web Ontology Language Manchester Syntax. W3C Candidate Recommendation (October 2009), http://www.w3.org/TR/owl2-manchester-syntax/
4. OWL 2 Web Ontology Language Structural Specification and Functional-style Syntax. W3C Candidate Recommendation (October 2009), http://www.w3.org/TR/owl2-syntax/
5. The Protégé ontology editor and knowledge-base framework (Version 4.1) (July 2011), http://protege.stanford.edu/
6. Alechina, N., Logan, B., Nga, N.H., Rakib, A.: Verifying Time and Communication Costs of Rule-Based Reasoners. In: Peled, D.A., Wooldridge, M.J. (eds.) MoChArt 2008. LNCS, vol. 5348, pp. 1–14. Springer, Heidelberg (2009)
7. Alechina, N., Logan, B., Nguyen, H.N., Rakib, A.: Automated Verification of Resource Requirements in Multi-Agent Systems Using Abstraction. In: van der Meyden, R., Smaus, J.-G. (eds.) MoChArt 2010. LNCS, vol. 6572, pp. 69–84. Springer, Heidelberg (2011)
8. Bechhofer, S., van Harmelen, F., Hendler, J.A., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference, World Wide Web Consortium, recommendation rec-owl-ref-20040210 (February 2004)
9. Broadfield, L., Banerjee, S., Jewers, H., Pollett, A.J., Simpson, J.: Guidelines for the management of cancer-related pain in adults. Supportive care cancer site team, cancer care Nova Scotia, Canada (2005)
10. Clavel, M., Eker, S., Lincoln, P., Meseguer, J.: Principles of Maude. Electronic Notes in Theoretical Computer Science 4, 65–89 (1996)
11. Cost, R.S., Finin, T.W., Joshi, A., Peng, Y., Nicholas, C.K., Soboroff, I., Chen, H., Kagal, L., Perich, F., Zou, Y., Tolia, S.: ITtalks: A case study in the Semantic Web and DAML+OIL. IEEE Intelligent Systems 17, 40–47 (2002)
12. Culbert, C.: CLIPS reference manual. NASA (2007)
13. Dekhtyar, M.I., Dikovsky, A.J., Valiev, M.K.: On complexity of verification of interacting agents' behavior. Annals of Pure and Applied Logic 141(3), 336–362 (2006)
14. Ferris, F.D., Balfour, H.M., Bowen, K., Farley, J., Hardwick, M., Lamontagne, C., Lundy, M., Syme, A., West, P.J.: A model to guide hospice palliative care: Based on national principles and norms of practice (March 2002)
15. Friedman-Hill, E.J.: Jess, the rule engine for the java platform. Sandia national laboratories (2008)
16. Glimm, B., Horridge, M., Parsia, B., Patel-Schneider, P.F.: A syntax for rules in OWL 2. In: Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009), vol. 529. CEUR (2009)
17. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: Combining logic programs with description logic. In: Proceedings of the 12th International Conference on World Wide Web, pp. 48–57. ACM Press (2003)
18. Gruber, T.: A translation approach to protable ontology specifications. Knowledge Acquisition 5, 199–220 (1993)
19. Hardiker, N., Coenen, A.: A formal foundation for ICNP. Journal of Stud. Health Technol. Inform. 122, 705–709 (2006)
20. Hendler, J.: Agents and the semantic web. IEEE Intelligent Systems 16, 30–37 (2001)

oarieli@mta.ac.il

21. Horridge, M., Bechhofer, S.: The OWL API: A java API for working with OWL 2 Ontologies. In: 6th OWL Experienced and Directions Workshop (OWLED 2009) (October 2009)
22. Horrocks, I., Patel-Schneider, P.F.: A proposal for an OWL rules language. In: Proceedings of the 13th International Conference on World Wide Web, WWW 2004, pp. 723–731. ACM Press (2004)
23. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web rule language combining OWL and RuleML. Acknowledged W3C submission, standards proposal research report: Version 0.6 (April 2004)
24. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. Journal of Web Semantics 3(2-3), 79–115 (2005)
25. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for owl: nominal schemas for integrating rules and ontologies. In: Proceedings of the 20th International Conference on World Wide Web, pp. 645–654. ACM (2011)
26. Krötzsch, M., Mehdi, A., Rudolph, S.: Orel: Database-driven reasoning for OWL 2 profiles. In: Proceedings of the 23rd International Workshop on Description Logics, vol. 573. CEUR (2010)
27. McGuinness, D., Smith, M., Welty, C.: OWL Web Ontology Language Guide, W3C Recommendation (February 2004), http://www.w3.org/TR/owl-guide/
28. Motik, B., Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language: Profiles, W3C Recommendation (October 2009), http://www.w3.org/TR/owl2-profiles/
29. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 3, 41–60 (2005)
30. Mousavi, A., Nordin, M.J., Othman, Z.A.: An ontology driven, procedural reasoning system-like agent model,for multi-agent based mobile workforce brokering systems. Journal of Computer Science 6, 557–565 (2010)
31. Rabbi, F., Mashiyat, A., MacCaull, W.: Model checking workflow monitors and its application to a pain management process. In: Proceedings of International Symposium on Foundations of Health Information Engineering and Systems, Johannesburg, South Africa, pp. 110–127 (2011)
32. Rakib, A.: Verifying requirements for resource-bounded agents. Ph.D. thesis, The University of Nottingham (2011)
33. Rao, A.S., Georgeff, M.P.: BDI Agents: From Theory to Practice. In: Proceedings of the First International Conference on Multi-agent Systems, pp. 312–319. The MIT Press (1995)
34. Ruan, J., MacCaull, W.: Data-aware monitoring for healthcare workflows using formal methods. In: Proceedings of the Second Workshop Knowledge Representation for Health Care (KR4HC 2010), Lisbon, Portugal, pp. 51–60 (2010)
35. Subercaze, J., Maret, P.: SAM - semantic agent model for swrl rule-based agents. In: Proceedings of the International Conference on Agents and Artificial Intelligence, pp. 245–248. INSTICC Press (2010)
36. Subrahmanian, V.S., Bonatti, P., Dix, J., Eiter, T., Kraus, S., Ozcan, F., Ross, R.: Heterogeneous Agent Systems. MIT Press (2000)
37. Tzafestas, S.G.: Knowledge-Based System Diagnosis, Supervision, and Control. Plenum Publishing Co. (1988)

oarieli@mta.ac.il

Formalizing Production Systems with Rule-Based Ontologies

Martín Rezk[1] and Michael Kifer[2]

[1] KRDB Research Center, Free University of Bozen-Bolzano, Bolzano, Italy
rezk@inf.unibz.it
[2] Department of Computer Science, Stony Brook University, NY 11794-4400, U.S.A.
kifer@cs.stonybrook.edu

Abstract. In this paper we proposed a new semantics for the combination of production systems with *arbitrary DL* ontologies. Unlike previous approaches, the semantics presented here allow looping rules and can handle inconsistencies produced by the interaction of the rule actions and the ontology. We also define a sound embedding of such semantics, restricted to rule-based *DL* Ontologies, into Transaction Logic with partial action definitions ($\mathcal{TR}^{PAD}$). This reduction adds a declarative semantics to the combination. To model production systems in $\mathcal{TR}^{PAD}$, we extend $\mathcal{TR}^{PAD}$ with default negation and define the well-founded semantics for it.

Keywords: Transaction Logic, Well-founded semantics, Ontologies, Production Systems, Knowledge Representation.

1 Introduction

Production systems (PS) are one of the oldest knowledge representation paradigms that are still popular today. Production systems are widely used in biomedical information systems, to enforce constraints on databases, to model business processes, accounting, etc.

Such systems consist of a set of *production rules* that rely on forward chaining reasoning to update the underlying database, called *working memory*. Traditionally, PS have had only operational semantics, where satisfaction of rule conditions is checked using pattern matching, and rule actions produce assertion and deletions of facts from the working memory. PS syntax and semantics have been standardized as W3C's Production Rule Dialect of the Rule Interchange Format (RIF-PRD) [14]. The RIF-PRD specification has a number of limitations, however. First, it omits certain important primitives that are found in many commercial production systems such as IBM's *JRules* [19]. The *FOR*-loop and the *while*-loop constructs are examples of such an omission. Second, RIF-PRD still does not integrate with ontologies [3,18].

To illustrate the need for ontology integration, consider a set PS that keeps a number of clinical databases that are compliant with the health insurance regulations. The clinical record of each patient together with other data must be accessible by all the clinics in the network. This needs a shared vocabulary

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 332–351, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

that, in this case, is defined in a shared *DL* ontology. However, each PS can have extra concepts outside the ontology, which are meant for local use only. The following production rules state that *(i)* if a doctor D requests a DNA test T to be performed for patient P, then the system records that P is taking the test T; *(ii)* if a patient gets cured, then she cannot be unhealthy; and *(iii)* if a patient receives medicine, then she gets cured.

r_1 : `Forall` D, P, T : `if` $\mathsf{requested}(D, P, T) \wedge \mathsf{dnaT}(T)$ `then` $\mathit{Assert}(\mathsf{takesT}(P, T))$
r_2 : `For` P : $\mathsf{cured}(P)$ `do` $\mathit{Retract}(\mathbf{neg}\,\mathsf{healthy}(P))$
r_3 : `Forall` P : `if` $\mathsf{rcv_meds(P)}$ `then` $\mathit{Assert}(\mathsf{cured}(P))$

The *DL* ontology that defines the shared concepts and implements different constraints is as follows

$$\mathsf{flu} \sqsubseteq \mathbf{neg}\,\mathsf{healthy} \qquad \mathsf{dnaT} \sqsubseteq \mathbf{neg}\,\mathsf{virusT} \qquad \exists \mathsf{takesT}.\mathbf{neg}\,\mathsf{virusT} \sqsubseteq \mathsf{healthy}$$

The *DL* axioms say that a patient with a flu is not a healthy patient, that DNA tests do not search for viruses, and that if a person is taking a test not related with any virus disease, then we can conclude that she is healthy. Here we are using explicit negation **neg** [22] to say that patients taking the DNA test should not be considered unhealthy. This type of negation is the preferred way of adding explicit negative information in rule-based knowledge representation; it is weaker than classical negation, does not add complexity to the logic, and makes knowledge representation more natural. The `Forall` construct in r_1 should not be confused with the `For` construct in r_2. The former is just a way RIF-PRD declares variables used in the body of a rule. The latter is a FOR-loop extension found in commercial systems, but not in RIF-PRD.

The complexity of the regulations in our example makes it difficult to determine whether executing a production rule leaves the database in a compliant state. Suppose we have the following database $\mathsf{WM}_0 = \{\mathsf{requested}(\text{Smith}, \text{Laura}, \text{pcr}), \mathsf{flu}(\text{Laura}), \mathsf{dnaT}(\text{pcr}), \mathsf{rcv_meds(Laura)}\}$.

This example raises several question: *(i)* Suppose that we execute r_3 with P instantiated with *Laura*. How do we interpret the retraction executed by r_2, (with P again instantiated with *Laura*) given that $\mathbf{neg}\,\mathsf{healthy}(\text{Laura})$ is inferred by the ontology, *(ii)* how to interpret the rule conditions of r_1 and r_2 given the open world semantics of *DL*, and *(iii)* how do we treat the inconsistency that results after execution of rule r_1 in WM_0? (Observe that in the state resulting from execution of r_1 in WM_0 we can infer $\mathsf{healthy}(\text{Laura})$ and $\mathbf{neg}\,\mathsf{healthy}(\text{Laura})$.)

To answer these questions we need to define a precise semantics (both model-theoretic and computational) to the combination of rules, ontologies, and production systems,

Our contribution in this paper is three-fold: *(i)* a new semantics for production systems augmented with DL ontologies that includes looping-rules, and can handle inconsistency; *(ii)* a sound embedding of the combination of PS and rule-based ontologies into *Transaction Logic with partially defined actions* (abbr., $\mathcal{TR}^{PAD}$) [25], which provides a model-theoretic semantics to the combination; *(iii)* an extension of $\mathcal{TR}^{PAD}$ with default negation under a variant of the well-founded semantics [29] for $\mathcal{TR}^{PAD}$.

oarieli@mta.ac.il

Transaction logic [5,6,7] was chosen because it provides a natural platform that satisfies most of the requirements to model the combination of PS and ontologies. It was designed from the outset as a formalism for declarative specification of complex state-changing transactions in logic programming, and it has been successfully used for planning [6], knowledge representation [8], active databases [6], event processing [1], workflow management and Semantic Web services [11,12,27,28], and as a declarative alternative to non-logical features in Prolog, like the *assert* and *retract* operators [7], which are also present in production systems.

Our formalization is significantly more general than RIF-PRD or other existing formalizations of production rules in that it supports wider ontology integration and covers important extensions that exist in commercial systems such as the aforesaid *FOR*-loop.

This paper is organized as follows. Section 2 briefly surveys previous results on the combination of PS and ontologies, and on the reduction of PS to formalisms with denotational semantics. Section 3 presents the necessary background on first order logic and description logic. Section 4 introduces an operational semantics for production systems augmented with DL ontologies. Section 5 augment $\mathcal{TR}^{PAD}$ with default negations, and provides a well-founded semantics for such extension. Section 6 provides a reduction from the semantics proposed here to $\mathcal{TR}^{PAD}$ and presents soundness results for this reduction. Section 7 concludes the paper. Proofs of the main results and further details are found in [24].

2 Related Work

In this section we compare our approach with other literature on the declarative semantics for production systems and on the operational and declarative semantics for the combination of PS and ontologies. The work described in [26,13] provides an operational and model-theoretic semantics to the combination of PS and ontologies. The model-theoretic semantics is given by an embedding of PS into fix-point logic. However, they cannot handle looping rules, their semantics cannot handle inconsistencies, their interpretation of retraction of *DL* facts is not intuitive since a fact can remain true after being deleted, and their reduction to a declarative formalism is considerably more complex than the one presented here. In [21,31], the goal is to devise languages for unifying some aspects of active rules, logic rules, and production systems. They do not deal with considerably more complex standard languages such as production systems augmented with ontologies and looping rules. In particular, [21,31] do not show how to embed production systems into those languages, although they provide some examples showing how typical production rules can be expressed in their language. In [23] the authors only allow a very restricted type of production systems: stratified PS. Such PS are much weaker that the ones formalized here, and again, they do not consider ontologies. In addition, they do not tackle the problem of the integration with ontologies. In [10,4], the authors reduce the semantics of PS to logic programming (LP). Their reduction is considerably more complex and less

oarieli@mta.ac.il

compact than ours—it results in an infinite number of rules. In addition, they use stable models semantics which has much higher computational complexity than the well founded semantics used here. Given the complexity of such a reduction, the proposed integration with LP ontologies is not ideal, since the ontology needs to be transformed with state arguments and auxiliary predicates. In addition, neither of them allow looping rules. Finally, [20] presents a new formalism that combines some aspects of logic rules and production rules. However, negation in rule conditions[1] and looping rules are disallowed. Furthermore, their embedding into Horn Logic is less clear and compact than our embedding in $\mathcal{TR}^{PAD}$.

3 Preliminaries

In this Section we briefly review the basic notions from Description Logic (*DL*) that we will use throughout the paper. Details can be found in [3].

Description Logic is a family of knowledge representation formalisms that provide a syntax and a model-theoretic semantics for a compact representation of information. The alphabet of a *DL* language $\mathcal{L}$ includes a countably infinite disjoint sets of variables $\mathcal{V}$, constant symbols $\mathcal{C}$, and unary and binary predicate symbols (concepts and roles respectively) $\mathcal{P}$. $\mathcal{L}$ includes the logical connectives $\sqcup, \sqcap, \neg, \forall, \exists, \sqsubseteq$.

A *DL* knowledge base has two parts: the *TBox*, with terminological knowledge, which consists of a number of concept axioms, and the *ABox*, which consists of assertions about actual individuals. Concept axioms in the TBox are of the form $C \sqsubseteq D$, meaning the extension of C is a subset of the extension of D. Concepts and TBox axioms can be understood as formulas of first-order logic with one free variable and closed universal formulas respectively. Therefore, the semantics of *DL* can be given by its translation to FOL. Details can be found in [3]. To integrate ontologies with production systems, we will later start using Herbrand domains and the unique name assumption (UNA). UNA is a commonly made assumption in *DL*s literature as well [2].

Therefore, we will use the Herbrand semantics from the outset. As is well-known, this semantics is equivalent to the general one for universal clausal form.

The semantics defines *semantic structures*. The domain of a Herbrand semantic structure is called the Herbrand universe $\mathcal{U}$; in our restricted case it is just the set of all constants $\mathcal{C}$ in the language $\mathcal{L}$. The Herbrand base $\mathcal{B}$ is a set of all ground literals in the language.

Definition 1 (Semantic Structure). *A* semantic structure $\mathcal{I}$ *is a triple* $\langle \mathcal{U}, \mathbf{B}, \sigma \rangle$*, where*

- $\mathcal{U}$ *is the Herbrand universe.*
- $\mathbf{B}$ *is a subset of* $\mathcal{B}$*.*
- σ *is a* ***variable assignment****, i.e., a mapping* $\mathcal{V} \longrightarrow \mathcal{U}$*.* □

[1] The authors informally claim that negation could be added, but they do not provide formal details.

oarieli@mta.ac.il

The definitions of satisfaction and entailment are as usual.

In Section 6, we will use *DL*s that can be embedded into Logic Programming (LP). In particular, [17] defines a class of *DL*s called ***Datalog-rewritable*** DL*s*. This class is interesting in our setting because reasoning with Datalog-rewritable *DL*s can be reduced to reasoning with Datalog programs. Due to space limitation, we will omit the details of these *DL*s. Complete definitions and the relationship with OWL can be found in [17].

4 Production Systems Augmented with Ontologies

In this section we propose a new semantics for the combination of production systems and *arbitrary DL* ontologies. This approach follows the outline of [26], but includes looping rules, it can handle inconsistencies produced by the system, and it gives a more intuitive semantics to the retraction of *DL* facts.

4.1 Syntax

The alphabet of a language $\mathcal{L}_{\mathsf{PS}}$ for a production system is defined the same way as in the case of *DL* except that now the set of all predicates $\mathcal{P}$ is partitioned into two countably infinite subsets, $\mathcal{P}_{PS}$ and $\mathcal{P}_{DL}$. The latter will be used to represent predicates occurring in the ontology. A term is either a variable or a constant symbol and, to avoid unnecessary distractions, we will leave out the various additional forms allowed in RIF, such as frames and the RIF membership and subclass relations ($o\#t$, $t\#\#s$). However, they can easily be added without increasing the complexity of the problem. A **atomic formula** is a statement of the form $p(t_1 \dots t_n)$, where $p \in \mathcal{P}$. A literal is either an atom, a formula of the form $\mathbf{neg}\, f$ where f is a $\mathcal{P}_{DL}$-atom, or a formula of the form $\neg f$ where f is a $\mathcal{P}_{PS}$-atom.

A **condition formula** has one of the following forms: a literal l, $\phi_1 \wedge \phi_2$ or $\phi_1 \vee \phi_2$ where ϕ_1 and ϕ_2 are condition formulas. Observe that all the rule conditions in our example are condition formulas. An **atomic action** is a statement that has one the following forms:

- ***assert***(l): Adds the literal l to the working memory
- ***retract***(l) : $\begin{cases} \text{if } p \in \mathcal{P}_{PS} & \text{Removes the } atom^2 l \text{ from the working memory} \\ \text{if } p \in \mathcal{P}_{DL} & \text{Enforces the literal } l \text{ to be false in the working memory} \end{cases}$ □

Beside these elementary actions, RIF also provides actions to change or delete objects and properties. Such actions can be treated similarly to `FOR`-rules below or as sequences of simpler actions, so we leave them out as well.

Definition 2 (Production System Augmented with Ontology). *A* production system augmented with ontology *(abbr.,* production system*, or* PS*) is a tuple* $\mathsf{PS} = (\mathcal{T}, \mathsf{L}, R)$ *such that*

[2] Negative literals with predicate symbols in $\mathcal{P}_{PS}$ cannot occur in the working memories. See Definition 3.

oarieli@mta.ac.il

– $\mathcal{T}$ *is a* DL *ontology (TBox) whose predicates belong to* $\mathcal{P}_{\mathrm{DL}}$*;*
– L *is a set of rule labels, and*
– R *is a set of rules, which are statements of one of the following forms*[3]

$$\texttt{IF-THEN Rule:} \quad r\colon \texttt{Forall}\ \boldsymbol{x}\colon \texttt{if}\ \ \phi_r(\boldsymbol{x})\ \texttt{then}\ \psi_r(\boldsymbol{x}) \tag{1}$$

$$\texttt{FOR Rule:} \quad r\colon\ \ \texttt{For}\ \boldsymbol{x}\colon \phi_r(\boldsymbol{x})\ \ \texttt{do}\ \psi_r(\boldsymbol{x}) \tag{2}$$

where (i) $r \in \mathsf{L}$ *is the above rule's label, (ii)* ϕ_r *is a condition formula in* $\mathcal{L}$ *with free variables* $\boldsymbol{x}$*, and (iii)* $\psi_r(\boldsymbol{x})$ *is a sequence of atomic actions with free variables contained in* $\boldsymbol{x}$*.* □

4.2 Operational Semantics

We now turn to the operational semantics of the combination of PS with ontologies. In a PS, two different constants represent two different domain elements, which is called the *unique name assumption*. In addition, production systems assume that each constant symbol is also a symbol in the domain of discourse, i.e., they are dealing with Herbrand domains. It is also worth noting that the semantics presented in this section does not depend on the specifics of the DL associated with production systems.

Definition 3 (Working Memory). *A* ***working memory***, WM, *for a PS language* $\mathcal{L}$ *is a disjoint union* $\mathsf{WM} = \mathsf{WM}_{PS} \uplus \mathsf{WM}_{\mathrm{DL}}$ *where* WM_{PS} *is a set of ground atoms that use predicate symbols from* $\mathcal{P}_{PS}$ *and* $\mathsf{WM}_{\mathrm{DL}}$ *is a set of ground literals that use predicate symbols from* $\mathcal{P}_{\mathrm{DL}}$*.* □

Definition 4 ($\mathcal{T}$-structure). *Let* $\mathcal{T}$ *be a* DL *TBox. A* $\mathcal{T}$**-structure**, $\mathcal{I}$*, for a PS language* $\mathcal{L}_{\mathsf{PS}}$ *has the form*

$$\mathcal{I} = (\mathsf{WM}_{PS} \uplus \mathsf{WM}_{\mathrm{DL}} \uplus \mathbf{E}_{\mathrm{DL}}, \sigma)$$

where $\mathsf{WM} = \mathsf{WM}_{PS} \uplus \mathsf{WM}_{\mathrm{DL}}$ *is a working memory,* $\mathbf{E}_{\mathrm{DL}}$ *is a set of* $\mathcal{P}_{\mathrm{DL}}$*-literals,* σ *is a variable assignment, and* $(\mathsf{WM}_{\mathrm{DL}} \uplus \mathbf{E}_{\mathrm{DL}}, \sigma)$ *is a model of* $\mathcal{T}$*.* □

We say that (WM, σ), where WM is a working memory, is a ***prestructure***.

Example 1. In our running example, the two disjoint sets composing the initial working memory WM_0 are as follows:

$$\mathsf{WM}_{0\,PS} = \{\mathsf{requested}(\mathrm{Smith}, \mathrm{Laura}, \mathrm{pcr}),\ \mathsf{rcv_meds}(\mathrm{Laura})\}.$$
$$\mathsf{WM}_{0\,DL} = \{\mathsf{flu}(\mathrm{Laura}), \mathsf{dnaT}(\mathrm{pcr})\}$$

In addition, we can build up a $\mathcal{T}$-structure, $\mathcal{I}$, by pairing any arbitrary assignment σ with WM_0 together with $\{\mathbf{neg}\,\mathsf{healthy}(\mathrm{Laura})\}$. That is, $\mathcal{I} = (\mathsf{WM}_0 \uplus \{\mathbf{neg}\,\mathsf{healthy}(\mathrm{Laura})\}, \sigma)$. □

[3] To avoid a misunderstanding, recall that the `Forall` construct is just a RIF-PRD syntax for declaring variables; it does not indicate a loop. In contrast, the `For-do` construct specifies a loop; it is found only in commercial PS systems, like JRules.

oarieli@mta.ac.il

Definition 5 (Satisfaction). *A $\mathcal{T}$-structure $\mathcal{I} = (\mathsf{WM}_{PS} \uplus \mathsf{WM}_{\mathrm{DL}} \uplus \mathbf{E}_{\mathrm{DL}}, \sigma)$ satisfies a literal l, denoted $\mathcal{I} \models l$, iff*

- *if l is a $\mathcal{P}_{\mathsf{PS}}$-atom then $l^{\mathcal{I}} \in \mathsf{WM}_{PS}$*
- *if l is a $\mathcal{P}_{\mathrm{DL}}$-literal then $\mathsf{WM}_{\mathrm{DL}} \uplus \mathbf{E}_{\mathrm{DL}} \models l^{\mathcal{I}}$*

*If ϕ is a formula of the form $\neg\phi_1$, $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$ then we define $\mathcal{I} \models \phi$ as usual in FOL. A formula ϕ **holds** in a prestructure (WM, σ) relative to an ontology $\mathcal{T}$, denoted $\mathcal{T}, (\mathsf{WM}, \sigma) \models \phi$, iff $\mathcal{I} \models \phi$ for every $\mathcal{T}$-structure of the form $\mathcal{I} = (\mathsf{WM} \uplus \mathbf{E}_{\mathrm{DL}}, \sigma)$ (That is, WM and σ are fixed but the $\mathbf{E}_{\mathrm{DL}}$ varies.)* □

Example 2. Consider again the initial working memory WM_0 from our running example, and let (WM_0, σ) be a prestructure. Observe that *(i)* the formula $\neg\mathsf{requested}(\mathrm{Smith}, \mathrm{Laura}, \mathrm{pcr})$ holds in (WM_0, σ) but *(ii)* $\neg\mathsf{takeT}(\mathrm{Laura}, \mathrm{pcr})$ does not. We can conclude *(i)* because $\mathsf{requested}(\mathrm{Smith}, \mathrm{Laura}, \mathrm{pcr})$ is a $\mathcal{P}_{PS}$ atom and it does not belong to WM_0. On the other hand, *(ii)* follows since $\neg\mathsf{takeT}(\mathrm{Laura}, \mathrm{pcr})$ is a $\mathcal{P}_{DL}$ atom and there is a $\mathcal{T}$-structure with working memory WM_0 and assignment σ that satisfies $\neg\mathsf{takeT}(\mathrm{Laura}, \mathrm{pcr})$. Note that $\mathbf{neg}\,\mathsf{takeT}(\mathrm{Laura}, \mathrm{pcr})$ does not hold in (WM_0, σ) either. □

A prestructure is $\mathcal{T}$-***consistent*** if there is a $\mathcal{T}$-structure with the same working memory and variable assignment, i.e., $(\mathsf{WM} \uplus \mathbf{E}_{DL}, \sigma)$, that does not entail f and $\mathbf{neg}\, f$ for any atom f. A working memory is $\mathcal{T}$-consistent if it is part of a $\mathcal{T}$-consistent prestructure.

Definition 6 (Atomic Transition). *Let (WM, σ) be a prestructure, t_1, t_2 be terms, and α be an action. We say that there is an α-**transition** from (WM, σ) to (WM', σ), denoted $(\mathsf{WM}, \sigma) \overset{\alpha}{\twoheadrightarrow} (\mathsf{WM}', \sigma)$, iff*

- *if $\alpha = \boldsymbol{assert}(p(t_1, t_2))$ then $\mathsf{WM}' = (\mathsf{WM} \cup \{p(t_1^\sigma, t_2^\sigma)\}) \setminus \{\mathbf{neg}\, p(t_1^\sigma, t_2^\sigma)\}$*
- *if $\alpha = \boldsymbol{retract}(p(t_1, t_2)])$ then* $\begin{cases} \text{if } p \in \mathcal{P}_{PS} & \mathsf{WM}' = \mathsf{WM} \setminus \{p(t_1^\sigma, t_2^\sigma)\} \\ \text{if } p \in \mathcal{P}_{\mathrm{DL}} & \mathsf{WM}' = (\mathsf{WM} \cup \\ & \{\mathbf{neg}\, p(t_1^\sigma, t_2^\sigma)\}) \setminus \{p(t_1^\sigma, t_2^\sigma)\} \end{cases}$

where t^σ is $\sigma(t)$ if t is a variable and it is t if t is a constant. □

In the remainder, we will write $(\mathsf{WM}_0, \sigma) \overset{\alpha_1 \ldots \alpha_n}{\twoheadrightarrow} (\mathsf{WM}_n, \sigma)$, to denote the sequence of transitions:

$$(\mathsf{WM}_0, \sigma) \overset{\alpha_1}{\twoheadrightarrow} (\mathsf{WM}_1, \sigma) \overset{\alpha_2}{\twoheadrightarrow} (\mathsf{WM}_2, \sigma) \overset{\alpha_3}{\twoheadrightarrow} \ldots \overset{\alpha_{n-1}}{\twoheadrightarrow} (\mathsf{WM}_{n-1}, \sigma) \overset{\alpha_n}{\twoheadrightarrow} (\mathsf{WM}_n, \sigma)$$

If, for some σ and $n \geq 1$, there is a transition $(\mathsf{WM}_0, \sigma) \overset{\alpha_1 \ldots \alpha_n}{\twoheadrightarrow} (\mathsf{WM}', \sigma)$ between prestructures then we will also write $\mathsf{WM}_0 \overset{\alpha_1 \ldots \alpha_n}{\twoheadrightarrow} \mathsf{WM}'$.

Suppose that there is a transition of the form $\mathsf{WM} \overset{\alpha}{\twoheadrightarrow} \hat{\mathsf{WM}}$ and $\hat{\mathsf{WM}}$ is not $\mathcal{T}$-consistent. The $\mathcal{T}$-**consistent result** of applying α to WM is the intersection of all the maximal subsets of $\hat{\mathsf{WM}}$ that contain $\hat{\mathsf{WM}} \setminus \mathsf{WM}$. This approach is known in the belief revision literature as *When in Doubt Throw it Out* (WIDTIO) [30]. This form of belief revision is in line with traditional ontologies and it has been also used in the context of evolution of *DL* knowledge bases [9].

oarieli@mta.ac.il

Example 3. Suppose we execute r_1 in WM_0. We obtain the inconsistent working memory $\mathsf{WM}_1 = \{\mathsf{requested}(\text{Smith}, \text{Laura}, \text{pcr}), \mathsf{takeT}(\text{Laura}, \text{pcr}), \mathsf{flu}(\text{Laura}), \mathsf{dnaT}(\text{pcr}), \mathsf{rcv_meds}(\text{Laura})\}$. We have two maximal consistent subsets of WM_1:

- $\mathsf{WM}'_1 = \{\mathsf{takeT}(\text{Laura}, \text{pcr}), \mathsf{dnaT}(\text{pcr}), \mathsf{requested}(\text{Smith}, \text{Laura}, \text{pcr}), \mathsf{rcv_meds}(\text{Laura})\}$.
- $\mathsf{WM}'_1 = \{\mathsf{takeT}(\text{Laura}, \text{pcr}), \mathsf{flu}(\text{Laura}), \mathsf{requested}(\text{Smith}, \text{Laura}, \text{pcr}), \mathsf{rcv_meds}(\text{Laura})\}$.

Thus, the consistent result is:

$$\mathsf{WM}_1^{cons} = \{\mathsf{takeT}(\text{Laura}, \text{pcr}), \mathsf{requested}(\text{Smith}, \text{Laura}, \text{pcr}), \mathsf{rcv_meds}(\text{Laura})\}\} \quad \square$$

A **consistent transition**, denoted $(\mathsf{WM}_0, \sigma) \overset{\alpha}{\twoheadrightarrow} (\mathsf{WM}_1^{cons}, \sigma)$, is a transition where the result of applying α in (WM_0, σ) is replaced the $\mathcal{T}$**-consistent result** of that action.

The following definition formalizes the *conflict resolution strategy* for a given rule r.

Definition 7 (Fireability). *We say that a rule r is **fireable** in a prestructure (WM_0, σ) if and only if:*

- `IF-THEN`*: r is of the form (1), $\phi_r(\sigma(\boldsymbol{x}))$ holds in WM_0, and there is a $\mathcal{T}$-consistent transition of the form*

$$(\mathsf{WM}_0, \sigma) \overset{\psi_r(\sigma(\boldsymbol{x}))}{\twoheadrightarrow} (\mathsf{WM}_n, \sigma)$$

- `FOR`*: r is of the form (2) and there are prestructures $(\mathsf{WM}_0, \sigma_0), (\mathsf{WM}_1, \sigma_0), (\mathsf{WM}_1, \sigma_1) \ldots (\mathsf{WM}_n, \sigma_{n-1})$ such that there are $\mathcal{T}$-consistent transitions of the form*

$$\begin{array}{c} (\mathsf{WM}_0, \sigma_0) \overset{\psi_r(\sigma_0(\boldsymbol{x}))}{\twoheadrightarrow} (\mathsf{WM}_1, \sigma_0) \\ (\mathsf{WM}_1, \sigma_1) \overset{\psi_r(\sigma_1(\boldsymbol{x}))}{\twoheadrightarrow} (\mathsf{WM}_2, \sigma_1) \\ \vdots \\ (\mathsf{WM}_{n-1}, \sigma_{n-1}) \overset{\psi_r(\sigma_{n-1}(\boldsymbol{x}))}{\twoheadrightarrow} (\mathsf{WM}_n, \sigma_{n-1}) \end{array} \tag{3}$$

  where the following conditions hold:
  1. Looping*: r's condition holds in each prestructure $(\mathsf{WM}_i, \sigma_i)$ $(0 \leq i \leq n-1)$*
  2. No repetitions*: For every pair of assignments σ_i, σ_j $(j \neq i$ and $0 \leq j, i \leq n-1)$ we have that $\sigma_i \neq \sigma_j$. That is, assignments cannot be re-used in the same rule execution.*
  3. Termination*: There is no assignment σ such that it produces a $\mathcal{T}$-consistent transition from WM_n, and (WM_n, σ) satisfies r's condition.*

In both cases above we say that r **causes transition** *from WM_0 to WM_n and denote it as $\mathsf{WM}_0 \overset{r}{\hookrightarrow} \mathsf{WM}_n$.* $\square$

Recall that a PS applies rules in three steps: (1) pattern matching, (2) conflict resolution, (3) rule execution, and then it loops back to (1). So far we have

oarieli@mta.ac.il

described only the steps (1) and (3). The next series of definitions describes Step (2) and show how looping is modeled in the semantics. This semantics does not depend on any particular conflict resolution strategy so, for concreteness, in Step (2) we will simply randomly choose a fireable rule from the conflict resolution set.[4] Some other works [4,13] use the same strategy.

The ***transition graph***, $\mathcal{T}_{\mathsf{PS}}$, of a production system is a directed labeled graph, whose set of nodes is the set of all working memories. There is an edge between two nodes WM and WM', labeled with α, σ for some action α and variable assignment σ, iff $(\mathsf{WM}, \sigma) \overset{\alpha}{\twoheadrightarrow} (\mathsf{WM}', \sigma)$. We will use $\mathcal{P}_{\mathsf{WM}}$ to denote the set of all paths (sequences of WMs) in the graph $\mathcal{T}_{\mathsf{PS}}$ starting at WM.

Definition 8 (Run). *A path π in $\mathcal{P}_{\mathsf{WM}_0}$ is a* **run** *$\mathcal{R}$ for a production system* PS *iff π can be split in paths $\pi_1, \ldots, \pi_n$ and there are rules $r_1 \ldots r_n$ such that for each $i = 1 \ldots n$, $WM_{i,start} \overset{r_i}{\hookrightarrow} WM_{i,end}$, where $WM_{i,start}$ is the first element in π_i and $WM_{i,end}$ is its last. Note that this implies that every π_i is a $\mathcal{T}$-consistent transition.* □

5 Extending $\mathcal{TR}^{PAD}$ with Default Negation

Transaction Logic with Partially Defined Actions [25], $\mathcal{TR}^{PAD}$, is a logic for programming actions and reasoning about them. In this section, we extend $\mathcal{TR}^{PAD}$ with *default* negation (a.k.a. negation as failure). Default negation allows a logic system to conclude the negation of any atom that the system unsuccessfully finishes exploring all possible proofs.

The alphabet of the language $\mathcal{L}_{\mathcal{TR}}$ of $\mathcal{TR}^{PAD}$ is defined the same way as in the *DL* case except that now the set of all predicates $\mathcal{P}$ is further partitioned into two subsets, $\mathcal{P}_{fluents}$ and $\mathcal{P}_{actions}$. The former will be used to represent facts in database states and the latter for transactions that change those states. Querying a fluent can be viewed as an action that does not change the underlying database state. We also add new symbols, $\rhd$ and $\overset{a}{\rightsquigarrow}$, where a is an atom whose predicate symbol is in $\mathcal{P}_{actions}$. Terms are defined as usual in first order logic. States are referred to with the help of special constants called ***state identifiers***; these will be usually denoted by boldface lowercase letters $\mathbf{d}$, $\mathbf{d}_1$, $\mathbf{d}_2$.

The symbol **neg** will be used to represent the explicit negation and **not** will be used for the *default* negation. These two symbols are applicable to fluents only. A fluent literal is either an atomic fluent or it has one of the following negated forms: $\mathbf{neg}\, f$, $\mathbf{not}\, f$, $\mathbf{not}\ \mathbf{neg}\, f$, where f is an atomic fluent. Literals that do not mention **not** are said to be **not**-free.

Note that in the ontologies one can have both **neg**- and $\neg$-literals, while $\mathcal{TR}^{PAD}$ uses **neg**- and **not**-literals instead. This is because logic programming rules cannot use classical negation, while ontologies do not use default negation.

Like the original Transaction Logic, $\mathcal{TR}^{PAD}$ contains logical connectives from the standard FOL ($\wedge, \vee, \forall, \exists$,) plus two additional logical connectives: the ***serial conjunction***, $\otimes$, and the modal operator $\Diamond$ for *hypothetical* execution.

[4] Recall that the conflict resolution set contains all the rules that can be fired on a given working memory.

oarieli@mta.ac.il

Informally, a serial conjunction of the form $\phi \otimes \psi$ represents an action composed of an execution of ϕ followed by an execution of ψ. A ***hypothetical formula***, $\Diamond\phi$, represents an action where ϕ is *tested* whether it can be executed at the current state, but no actual state changes take place. For instance, the first part of the following formula

$$\Diamond(\mathit{insert}(\mathsf{infection}) \otimes \mathit{bill_insurance} \otimes \mathsf{has_paid}) \otimes \mathit{insert}(\mathsf{takesT})$$

is a hypothetical test to verify that the patient's insurance company will pay in case of an infection after the blood test. The actual blood test is only performed if the hypothetical test succeeds. In particular, we will use hypothetical executions to check—before firing a rule—that executing the action associated with such a rule will not produce an inconsistent state. In this paper we will assume that hypothetical formulas contain only serial conjunctions of literals.

$\mathcal{TR}^{PAD}$ consists of ***serial-Horn*** rules, ***partial action definitions*** (PADs), and certain statements about states and actions, which we call *premises*. The syntax for all this is shown below, where c is a **not**-free literal, $c_1, \ldots, c_n$ are *literals* (fluents or actions), f is a **not**-free fluent literal, b_1, b_2 are conjunctions of *fluent* literals or hypotheticals (**not**-literals are ok), b_3, b_4 are conjunctions of **not**-free *fluent* literals, $\mathbf{d}_0, \mathbf{d}_1 \ldots$ are identifiers, and a is an action atom. These actions will be used to encode the *assert* and *retract* actions in production rules, as well as the laws of inertia (a.k.a. frame axioms). Observe that in this paper we address the issue of how to express frame axioms, not the larger issue of the frame problem, which aims to encode one general principle of inertia, rather than developing particular frame axioms that are suitable in specific applications.

Rules		**Premises**	
(i) $c \leftarrow c_1 \otimes \cdots \otimes c_n$	(a serial-Horn rule)	(iii) $\mathbf{d}_0 \rhd f$	(a **state**-premise)
(ii) $b_1 \otimes a \otimes b_2 \rightarrow b_3 \otimes a \otimes b_4$	(a PAD)	(iv) $\mathbf{d}_1 \overset{a}{\rightsquigarrow} \mathbf{d}_2$	(a **run**-premise)

The serial-Horn rule (i) is a statement that defines the literal c, which can be viewed as a calling sequence for a complex transaction and $c_1 \otimes \cdots \otimes c_n$ can be viewed as a definition for the actual course of action to be performed by that transaction. If c is a fluent literal then we require that c_1, ..., c_n are also fluents. In that case we call c a ***defined fluent*** and the rule itself a ***fluent rule***. Fluent rules are equivalent to regular Horn rules in logic programming. If c is an action, we will say that c is a ***compound action***, as it is defined by a rule. For instance, the serial-Horn rule $\mathit{r_1} \leftarrow \mathsf{requested}(D, P, T) \otimes \mathsf{dnaT}(T) \otimes \mathit{insert}(\mathsf{takesT}(P, T))$ defines a compound action *r_1*. This action behaves in the same way as the rule r_1 in our running example. The PAD (ii) means that if we know that b_1 holds before executing action a and b_2 holds after, we can conclude that b_3 must have held before executing a and b_4 must hold as a result of a. For instance, the PAD $(\mathsf{healthy}(P) \otimes \mathit{insert}(\mathsf{dnaT}(T))) \rightarrow (\mathit{insert}(\mathsf{dnaT}(T)) \otimes \mathsf{healthy}(P))$. states that if a patient is healthy, she remains so after adding a DNA type in the database. This is a simplified version of an inertial law in $\mathcal{TR}^{PAD}$. To sum up, we distinguish two kinds of actions: ***partially defined actions*** (abbr., *pda*) and ***compound actions***. Partially defined actions cannot be defined by rules—they are defined

oarieli@mta.ac.il

by PAD statements only. In contrast, compound actions are defined via serial-Horn rules but not by PADs. Note that *pda*s can appear in the bodies of serial-Horn rules that define compound actions (see *r_1* above) and, in this way, $\mathcal{TR}^{PAD}$ can create larger action theories by composing smaller ones in a modular way.

Premises are statements about the initial and the final database states (state premises) and about possible state transitions caused by partially defined actions (run-premises). For example, to represent the initial database in our example, we can use the state premises

$$\mathbf{d}_0 \rhd \mathsf{dnaT}(\text{pcr}) \qquad \mathbf{d}_0 \rhd \mathsf{requested}(\text{Smith}, \text{Laura}, \text{pcr}) \qquad \mathbf{d}_0 \rhd \mathsf{flu}(\text{Laura})$$

The run-premise $\mathbf{d}_0 \overset{\mathit{insert}(\mathsf{takeT}(\boldsymbol{t}))}{\rightsquigarrow} \mathbf{d}_1$ says that executing the *pda* $\mathit{insert}(\mathsf{takeT}(\boldsymbol{t}))$ in the state associated with $\mathbf{d}_0$ leads to the state represented by $\mathbf{d}_1$.

A ***transaction*** is a statement of the form $?\text{-}_{\mathbf{d}_0}\, \exists \bar{X} \phi$, where $\phi = l_1 \otimes \cdots \otimes l_k$ is a serial conjunction of literals (both fluent and action literals) and $\bar{X}$ is a list of all the variables that occur in ϕ. Transactions in $\mathcal{TR}$ generalize the notion of queries in ordinary logic programming. For instance, $?\text{-}_{\mathbf{d}_0}\mathsf{flu}(\text{Laura}) \otimes r_1$ is a transaction that first checks if the patient has a flu in the initial state $\mathbf{d}_0$; if so, the compound action *r_1* is executed. If the execution of the transaction cannot proceed the already executed actions are undone and the underlying database state remains unchanged. This property is known as *atomicity* of transactions in databases. A $\mathcal{TR}^{PAD}$ ***transaction base*** is a set of serial-Horn rules. A $\mathcal{TR}^{PAD}$ ***action base*** is a set of PADs. A $\mathcal{TR}^{PAD}$ **specification** is a tuple $(\mathcal{E}, \mathbf{P}, \mathcal{S})$ where $\mathcal{E}$ is a $\mathcal{TR}^{PAD}$ action base, $\mathbf{P}$ is a $\mathcal{TR}^{PAD}$ transaction base, and $\mathcal{S}$ is a set of premises.

Semantics. Now we define a well-founded semantics [29] for $\mathcal{TR}^{PAD}$, which, to the best of our knowledge, has never been done before. This semantics uses three truth values, $\mathbf{u}$, $\mathbf{t}$ and $\mathbf{f}$, which stand for *true*, *false*, and *undefined* and are ordered as follows: $\mathbf{f} < \mathbf{u} < \mathbf{t}$. In addition, we will use the following operator $\sim$: $\sim \mathbf{t} = \mathbf{f}$, $\sim \mathbf{f} = \mathbf{t}$, $\sim \mathbf{u} = \mathbf{u}$. A **database state D** (or just a *state,* for short) is a set of ***ground*** (i.e., variable-free) fluent literals. In the language, states are referred to via state identifier constants, which were introduced earlier. The mapping between state identifiers and states is determined by *path structures*, defined next.

Definition 9 (Three-valued Partial Herbrand Interpretation). *A **partial Herbrand interpretation** is a mapping $\mathcal{H} : \mathcal{B} \mapsto \{\boldsymbol{f}, \boldsymbol{u}, \boldsymbol{t}\}$ that assigns a truth value, $\boldsymbol{f}, \boldsymbol{u}$, or $\boldsymbol{t}$, to every formula ϕ in $\mathcal{B}$.* □

A central feature in $\mathcal{TR}$ is the notion of *execution paths*, since $\mathcal{TR}$ formulas are evaluated over paths and *not* over states like in temporal logics [15].

Definition 10 (Three-valued Herbrand Path Structure). *A **Herbrand path structure** is a mapping $\mathbf{I}$ that assigns a partial Herbrand interpretation to every path. That is, for any path π, $\mathbf{I}(\pi)$ is an interpretation. So, for instance, $\mathbf{I}(\pi)(f)$ is a truth value for any literal f. This mapping must satisfy the restriction that for each ground base fluent f and database state $\mathbf{D}$:*

oarieli@mta.ac.il

$$\mathbf{I}(\langle\mathbf{D}\rangle)(f) = \boldsymbol{t} \;\; \textit{if } f \in \mathbf{D} \qquad \textit{and} \qquad \mathbf{I}(\langle\mathbf{D}\rangle)(\mathbf{neg}\, f) = \boldsymbol{t} \;\; \textit{if } \mathbf{neg}\, f \in \mathbf{D}$$

where $\langle\mathbf{D}\rangle$ *is a path that contains only one state,* $\mathbf{D}$*.*
In addition, $\mathbf{I}$ *includes a mapping of the form:*

$$\Delta_{\mathbf{I}} : \text{ State identifiers} \longrightarrow \text{ Database states}$$

which associates states (i.e., sets of atomic formulas) to state identifiers. We will usually omit the subscript. □

Intuitively, Herbrand path structures in $\mathcal{TR}$ play a role similar to transition functions in temporal logics by providing a link between states and actions.

An ***execution path*** of length k, or a ***k-path***, is a finite sequence of states, $\pi = \langle\mathbf{D}_1 \;\ldots\; \mathbf{D}_k\rangle$, where $k \geq 1$. A ***path abstraction*** is a finite sequence of state identifiers. If $\langle\mathbf{d}_1 \;\ldots\; \mathbf{d}_k\rangle$ is a path abstraction then $\langle\mathbf{D}_1 \;\ldots\; \mathbf{D}_k\rangle$, where $\mathbf{D}_i = \Delta(\mathbf{d}_i)$, is an execution path. We will also sometimes write $\mathcal{M}(\langle\mathbf{d}_1 \;\ldots\; \mathbf{d}_k\rangle)$ meaning $\mathcal{M}(\langle\Delta(\mathbf{d}_1) \;\ldots\; \Delta(\mathbf{d}_k)\rangle)$. A ***split*** of a path π is a pair of subpaths, π_1 and π_2, such that $\pi_1 = \langle\mathbf{D}_1 \ldots \mathbf{D}_i\rangle$ and $\pi_2 = \langle\mathbf{D}_i \ldots \mathbf{D}_k\rangle$ for some i $(1 \leq i \leq k)$. In this case, we write $\pi = \pi_1 \circ \pi_2$.

In the remainder of this section we will consider ground rules and PADs only. We can make this assumption without loosing generality because all the variables in a rule are considered to be universally quantified.

The following definition formalizes the idea that truth of $\mathcal{TR}$ formulas is defined on paths.

Definition 11 (Satisfaction). *Let* $\mathbf{I}$ *be a Herbrand path structure,* π *be a path,* f *a ground* **not** *-free literal, and* G*,* G_1*,* G_2 *ground serial goals. We define* ***truth valuations*** *with respect to the path structure* $\mathbf{I}$ *as follows:*

- $\mathbf{I}(\pi)(f)$ *was already defined as part of the definition of Herbrand path structures.*[5]
- $\mathbf{I}(\pi)(\phi \otimes \psi) = max\{min(\mathbf{I}(\pi_1)(\phi), \mathbf{I}(\pi_2)(\psi)) \mid \pi = \pi_1 \circ \pi_2\}$
- $\mathbf{I}(\pi)(G_1 \wedge G_2) = min(\mathbf{I}(\pi)(G_1), \mathbf{I}(\pi)(G_2))$
- $\mathbf{I}(\pi)(\mathbf{not}\, \phi) = \sim \mathbf{I}(\pi)(\phi)$[6]
- $\mathbf{I}(\pi)(\diamond\phi) = \begin{cases} max\{\mathbf{I}(\pi')(\phi) \mid \pi' \textit{ is a path that starts at } \mathbf{D}\} & \textit{if } \pi = \langle\mathbf{D}\rangle \\ \boldsymbol{f} & \textit{otherwise} \end{cases}$
- $\mathbf{I}(\pi)(f \leftarrow G) = \boldsymbol{t}$ *iff* $\mathbf{I}(\pi)(f) \geq \mathbf{I}(\pi)(G)$
- $\mathbf{I}(\pi)(b_1 \otimes \alpha \otimes b_2 \rightarrow b_3 \otimes \alpha \otimes b_4) = \boldsymbol{t}$ *iff* π *has the form* $\langle\mathbf{D}_1, \mathbf{D}_2\rangle$, $\mathbf{I}(\langle\mathbf{D}_1, \mathbf{D}_2\rangle)(\alpha) = \boldsymbol{t}$*, and the following holds:*

$$\begin{aligned} & min\{min\{\mathbf{I}(\langle\mathbf{D}_1\rangle)(f) \mid f \in b_1\}, min\{\mathbf{I}(\langle\mathbf{D}_2\rangle)(f) \mid f \in b_2\}\} \\ \leq \; & min\{min\{\mathbf{I}(\langle\mathbf{D}_1\rangle)(f) \mid f \in b_3\}, min\{\mathbf{I}(\langle\mathbf{D}_2\rangle)(f) \mid f \in b_4\}\} \end{aligned}$$

We write $\mathbf{I}, \pi \models \phi$ *and say that* ϕ *is* ***satisfied*** *on path* π *in the path structure* $\mathbf{I}$ *if* $\mathbf{I}(\pi)(\phi) = \boldsymbol{t}$*.* □

In addition, we assume that the language includes the distinguished propositional constants $\mathbf{t}^\pi$, and $\mathbf{u}^\pi$ for each $\mathcal{TR}$ path π. Observe that since there is an infinite number of paths, there is an infinite number of such constants. Informally, $\mathbf{t}^\pi$ ($\mathbf{u}^\pi$) is a proposition that has the truth value $\mathbf{t}$(respectively $\mathbf{u}$) only on the path π, and it is false on every other path.

[5] Here `max` is taken over a finite set of truth values $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$.
[6] Recall that $\sim \mathbf{t} = \mathbf{f}$, $\sim \mathbf{f} = \mathbf{t}$, $\sim \mathbf{u} = \mathbf{u}$.

oarieli@mta.ac.il

Definition 12 (Model). *A path structure,* $\mathbf{I}$*, is a* ***model of a formula*** ϕ *if* $\mathbf{I}, \pi \models \phi$ *for every path* π*. In this case we write* $\mathbf{I} \models \phi$*. A path structure,* $\mathbf{I}$*, is a model of a set of formulas if it is a model of every formula in the set. A path structure,* $\mathbf{I}$*, is a* ***model of a premise-statement*** σ *iff:*

- σ *is a run-premise of the form* $\boldsymbol{d}_1 \overset{\alpha}{\rightsquigarrow} \boldsymbol{d}_2$ *and* $\mathbf{I}, \langle \boldsymbol{d}_1 \boldsymbol{d}_2 \rangle \models \alpha$*; or*
- σ *is a state-premise* $\boldsymbol{d} \rhd f$ *and* $\mathbf{I}, \langle \boldsymbol{d} \rangle \models f$*.*

$\mathbf{I}$ *is a* ***model*** *of a specification* $(\mathcal{E}, \mathbf{P}, \mathcal{S})$ *if* $\mathbf{I}$ *is an path structure that satisfies every PAD in* $\mathcal{E}$*, every rule in* $\mathbf{P}$ *and every premise in* $\mathcal{S}$*.* □

Example 4. Consider our running example. Let us present the specification $\Lambda = (\mathcal{E}, \mathbf{P}, \mathcal{S})$, that intuitively encodes the ontology and part of the PS. A complete encoding will be described in Section 6. Assume that $\mathcal{S}$ contains the premises already introduced in the previous section. The transaction base $\mathbf{P}$ contains the following rules encoding the ontology

$$\mathbf{neg}\ \mathsf{virusT}(T) \leftarrow \mathsf{dnaT}(T) \qquad \mathsf{healthy}(P) \leftarrow \mathsf{takesT}(P,T), \mathbf{neg}\ \mathsf{virusT}(T)$$
$$\mathbf{neg}\ \mathsf{healthy}(P) \leftarrow \mathsf{flu}(P)$$

The action base $\mathcal{E}$ contains two PADs encoding the (simplified) inertia laws, and the definition of the action $\mathit{insert}_{\mathsf{takeT}}$.

$$\begin{aligned}
\mathsf{dnaT}(P) \otimes \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathbf{not}\ \mathsf{inconsistent} &\rightarrow \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathsf{dnaT}(P) \\
\mathsf{flu}(P) \otimes \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathbf{not}\ \mathsf{inconsistent} &\rightarrow \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathsf{flu}(P) \\
\mathsf{rcv_meds}(P) \otimes \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathbf{not}\ \mathsf{inconsistent} &\rightarrow \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathsf{rcv_meds}(P) \\
\mathit{insert}_{\mathsf{takeT}}(P,T) &\rightarrow \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathsf{takeT}(P,T)
\end{aligned}$$

From the premises and the rules in Λ, we can see that any path structure $\mathcal{I}$ that models Λ satisfies

$$\begin{aligned}
&\mathcal{I}(\mathbf{d}_0)(\mathsf{dnaT}(\mathrm{pcr})) = \mathbf{t} \quad && \mathcal{I}(\mathbf{d}_0\mathbf{d}_1)(\mathit{insert}_{\mathsf{takeT}}(\mathrm{Laura}, \mathrm{pcr})) = \mathbf{t} \\
&\mathcal{I}(\mathbf{d}_0)(\mathsf{flu}(\mathrm{Laura})) = \mathbf{t} \quad && \mathcal{I}(\mathbf{d}_0)(\mathsf{rcv_meds}(\mathrm{Laura})) = \mathbf{t}
\end{aligned}$$

Now take an interpretation, $\mathcal{I}_1$, such that $\mathcal{I}_1(\mathbf{d}_1)(\mathsf{inconsistent}) = \mathbf{f}$. From the PADs in $\mathcal{E}$ instantiated with pcr, Laura, and Smith, we can conclude that:

$$\begin{aligned}
&\mathcal{I}_1(\mathbf{d}_1)(\mathsf{dnaT}(\mathrm{pcr})) = \mathbf{t} \qquad && \mathcal{I}_1(\mathbf{d}_1)(\mathsf{takeT}(\mathrm{Laura}, \mathrm{pcr})) = \mathbf{t} \\
&\mathcal{I}_1(\mathbf{d}_1)(\mathsf{flu}(\mathrm{Laura})) = \mathbf{t} \qquad && \mathcal{I}(\mathbf{d}_1)(\mathsf{rcv_meds}(\mathrm{Laura})) = \mathbf{t}
\end{aligned}$$

and from the rules in the ontology it follows that $\mathcal{I}_1(\mathbf{d}_1)(\mathsf{healthy}) = \mathbf{t}$ and $\mathcal{I}_1(\mathbf{d}_1)(\mathbf{neg}\ \mathsf{healthy}) = \mathbf{t}$. Thus, $\mathbf{d}_1$ is inconsistent in $\mathcal{I}_1$. □

In classical logic programming based on three-valued models, given two Herbrand *partial* interpretations $\mathbf{N}_1$ and $\mathbf{N}_2$, we say that $\mathbf{N}_1 \leq^c \mathbf{N}_2$ iff all **not**-free literals that are true in $\mathbf{N}_1$ are true in $\mathbf{N}_2$ and all **not**-literals that are true in $\mathbf{N}_1$ are true in $\mathbf{N}_2$. In addition, we say that $\mathbf{N}_1 \preceq^c \mathbf{N}_2$ iff all **not**-free literals that are true in $\mathbf{N}_1$ are true in $\mathbf{N}_2$ and all **not**-literals that are true in $\mathbf{N}_2$ are true in $\mathbf{N}_1$.

Definition 13 (Order on Path Structures). *Let* $\mathbf{M}_1$ *and* $\mathbf{M}_2$ *be two Herbrand path structures, then:*

oarieli@mta.ac.il

– Information ordering: $\mathbf{M}_1 \leq \mathbf{M}_2$ *if for every path,* π*, it holds that* $\mathbf{M}_1(\pi) \leq^c \mathbf{M}_2(\pi)$.
– Truth ordering: $\mathbf{M}_1 \preceq \mathbf{M}_2$ *if for every path,* π*, it holds that* $\mathbf{M}_1(\pi) \preceq^c \mathbf{M}_2(\pi)$. □

These two orderings are considerably different. The *truth ordering* minimize the *amount of truth*, by minimizing the atoms that are true and maximizing the atoms that are false on each path. In contrast, the *information ordering* minimizes the amount of information by minimizing both the atoms that are true and false in each path.

Example 5. Consider a path structure $\mathcal{I}_2$ for the specification Λ in Example 4 that coincides with $\mathcal{I}_1$ in the path $\langle \mathbf{d}_0 \rangle$ but differs in $\langle \mathbf{d}_1 \rangle$ as follows:

$$\mathcal{I}_2(\mathbf{d}_1)(\mathsf{dnaT}(\mathrm{pcr})) = \mathbf{u} \qquad \mathcal{I}_2(\mathbf{d}_1)(\mathsf{inconsistent}) = \mathbf{u}$$
$$\mathcal{I}_2(\mathbf{d}_1)(\mathsf{flu}(\mathrm{Laura})) = \mathbf{u} \qquad \mathcal{I}_2(\mathbf{d}_1)(\mathsf{takeT}(\mathrm{Laura}, \mathrm{pcr}) = \mathbf{t}$$

It is not hard to see that $\mathcal{I}_2$ is also a model of Λ, and moreover $\mathcal{I}_2 \preceq \mathcal{I}_1$. □

Definition 14 (Least Model). *A model* $\mathbf{M}$ *of a specification* $(\mathcal{E}, \mathbf{P}, \mathcal{S})$ *is* ***minimal*** *with respect to* $\preceq$ *iff for any other model,* $\mathbf{N}$*, of* $(\mathcal{E}, \mathbf{P}, \mathcal{S})$*, if* $\mathbf{N} \preceq \mathbf{M}$ *then* $\mathbf{N} = \mathbf{M}$*. The* ***least*** *model of* $(\mathcal{E}, \mathbf{P}, \mathcal{S})$*, denoted LPM*$(\mathcal{E}, \mathbf{P}, \mathcal{S})$*, is a minimal model that is unique.* □

The definition of *quotient* is key to the notion of well-founded $\mathcal{TR}^{PAD}$ models. It is modeled after [16] with appropriate extensions for PADs.

By $\mathcal{TR}^{PAD}$**-quotient** of an specification $(\mathcal{E}, \mathbf{P}, \mathcal{S})$ **modulo** a path structure $\mathbf{I}$ we mean a new specification, $\frac{(\mathcal{E},\mathbf{P},\mathcal{S})}{\mathbf{I}}$, which is obtained from $(\mathcal{E}, \mathbf{P}, \mathcal{S})$ by first *(i)* replacing every literal of the form **not** b in $\mathbf{P} \cup \mathcal{E}$ with $\mathbf{t}^\pi$ or $\mathbf{u}^\pi$ for every path π such that **not** b is true (respectively undefined) in $\mathbf{I}(\pi)$, and then *(ii)* removing all the remaining rules and PADs that have a literal of the form **not** b in the body that is false in $\mathbf{I}(\pi)$.

Next, we give a constructive definition of *well-founded models* for $\mathcal{TR}^{PAD}$ specifications in terms of a consequence operator. As in the classical case, the consequence operator, Γ, for a $\mathcal{TR}^{PAD}$ specification is defined as:

$$\Gamma(\mathbf{I}) = \mathrm{LPM}(\frac{(\mathcal{E}, \mathbf{P}, \mathcal{S})}{\mathbf{I}})$$

Now, suppose $\mathbf{I}_\emptyset$ is the path structure that maps each path π to the empty Herbrand interpretation in which all atoms are undefined. The ordinal powers of the consequence operator Γ are then defined inductively as follows:

– $\Gamma^{\uparrow 0}(\mathbf{I}_\emptyset) = \mathbf{I}_\emptyset$
– $\Gamma^{\uparrow n}(\mathbf{I}_\emptyset) = \Gamma(\Gamma^{\uparrow n-1}(\mathbf{I}_\emptyset))$, if n is a successor ordinal
– $\Gamma^{\uparrow n}(\mathbf{I}_\emptyset)(\pi) = \bigcup_{j \leq n} \Gamma^{\uparrow j}(\mathbf{I}_\emptyset)(\pi)$, if n is a limit ordinal □

The operator Γ is monotonic with respect to the $\leq$-order when $(\mathcal{E}, \mathbf{P}, \mathcal{S})$ is fixed. Because of this, the sequence $\{\Gamma^{\uparrow n}(\mathbf{I}_\emptyset)\}$ has a least fixed point and is computable via transfinite induction.

oarieli@mta.ac.il

Lemma 1. *(see [24]) The operator Γ is monotonic with respect to the information order relation $\leq$ when $(\mathcal{E}, \mathbf{P}, \mathcal{S})$ is fixed. That is,*

$$\textit{If} \quad \mathbf{I} \leq \mathbf{I}' \quad \textit{then} \quad \Gamma(\mathbf{I}) \leq \Gamma(\mathbf{I}')$$

Definition 15 (Well-founded Model). *The **well-founded** model of a $\mathcal{TR}^{PAD}$ specification $(\mathcal{E}, \mathbf{P}, \mathcal{S})$, written $WFM((\mathcal{E}, \mathbf{P}, \mathcal{S}))$, is defined as a limit of the sequence $\{\Gamma^{\uparrow n}(\mathbf{I}_\emptyset)\}$.*
$WFM((\mathcal{E}, \mathbf{P}, \mathcal{S}))$ is, indeed, a model of $(\mathcal{E}, \mathbf{P}, \mathcal{S})$, as shown in [24]. □

Example 6. Consider the specification in Example 4 together with the following rule defining the fluent inconsistent.

$$\mathsf{inconsistent} \leftarrow \mathsf{healthy}(P), \ \ \mathbf{neg}\,\mathsf{healthy}(P)$$

In the specification $\frac{\Lambda}{\mathbf{I}_\emptyset}$, the sets $\mathbf{P}$, remain the same since they all are **not**-free. In $\mathcal{E}$, only the frame axioms change as follows

$$\mathsf{dnaT}(P) \otimes \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathbf{u}^\pi \rightarrow \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathsf{dnaT}(P)$$
$$\mathsf{flu}(P) \otimes \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathbf{u}^\pi \quad \rightarrow \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathsf{flu}(P)$$

Since $\frac{\Lambda}{\mathbf{I}_\emptyset}$ is **not**-free, it has a minimal model [24] $\Gamma^{\uparrow 1}(\mathbf{I}_\emptyset) = \mathcal{I}_1$. It follows from the construction of $\mathcal{I}_1$ that $\mathcal{I}_1(\langle \mathbf{d}_1 \rangle)(\mathsf{inconsistent}) = \mathbf{u}$. It is not hard to see that in the WFM of Λ, inconsistent is also undefined in $\mathcal{I}_1(\langle \mathbf{d}_1 \rangle)$. This is because the frame axioms are preventing the inconsistency from occurring, but it is still detected. Without the rules encoding the ontology, inconsistent would be false in $WFM(\langle \mathbf{d}_1 \rangle)$. □

Theorem 1. *(see [24]) $WFM((\mathcal{E}, \mathbf{P}, \mathcal{S}))$ is the least model of $(\mathcal{E}, \mathbf{P}, \mathcal{S})$.*

6 Declarative Semantics

In this section we present the reduction of production systems augmented with Datalog-rewritable ontologies to $\mathcal{TR}^{PAD}$. Given an alphabet $\mathcal{L}_{\mathsf{PS}}$ for a production system PS, the corresponding language $\mathcal{L}_{\mathcal{TR}}$ of the target $\mathcal{TR}^{PAD}$ formulation will consist of symbols for rule labels, constants, and predicates. In addition, $\mathcal{L}_{\mathcal{TR}}$ has the following symbols: *(i)* the *pdas* *add_used* and *clean_used*, and for every predicate $p \in \mathcal{L}_{\mathsf{PS}}$, *Ins_p*, and *del_p*; *(ii)* the compound action *act*; *(iii)* the defined fluent inconsistent, and for every rule label r the defined fluent fireable_r; *(iv)* the fluents inertial and used. Intuitively, the *pdas* *Ins_p*, and *del_p* above represent the actions ***assert*** and ***retract***. The *pdas* *add_used* and *clean_used*, and the fluent used, are used to keep track of the assignments that has already been used to instantiate a FOR-production rule. The compound action *act* represents a generic production rule. The defined fluent fireable_r is true if the condition of the rule r holds and the action produces no inconsistencies. The defined fluent inconsistent is true, if there is an inconsistency in the state. The fluent inertial is used to distinguish inertial from non-inertial fluents.

Let $\psi = \alpha_1 \ldots \alpha_n$ be a sequence of atomic actions and $\phi = f_1 \wedge \cdots \wedge f_n \wedge l_1 \ldots l_m$ be a conjunction of atoms (f_i) and negative literals (l_j). We use $\widehat{\psi}$ and $\widehat{\phi}$ to denote

oarieli@mta.ac.il

the $\mathcal{TR}$ -serial conjunctions $\widehat{\psi} = \widehat{\alpha_1} \otimes \cdots \otimes \widehat{\alpha_n}$ and $\widehat{\phi} = f_1 \wedge \cdots \wedge f_m \wedge \sim l_1 \wedge \cdots \wedge \sim l_m$, where

$$\widehat{\alpha_i} = \begin{cases} \mathit{Ins_}p(\boldsymbol{t}) & \text{if } \alpha_i = \boldsymbol{assert}(p(\boldsymbol{t})) \\ \mathit{del_}p(\boldsymbol{t}) & \text{if } \alpha_i = \boldsymbol{retract}(p(\boldsymbol{t})) \end{cases}$$

$$\sim l_j = \begin{cases} \mathbf{not}\, f(\boldsymbol{c}) & \text{if } l_j = \neg f(\boldsymbol{c}) \wedge f \in \mathcal{P}_{PS} \\ \mathbf{neg}\, f(\boldsymbol{c}) & \text{if } l_j = \mathbf{neg}\, f(\boldsymbol{c}) \wedge f \in \mathcal{P}_{DL} \end{cases}$$

In the following, let $\mathsf{PS} = (\mathcal{T}, \mathsf{L}, R)$ be a production system. For simplicity we assume that conditions in production rules are conjunctions of fluent literals. In addition, we assume we have an ***initial working memory***, WM_0, that represents the knowledge we have about the initial state of the system.

The reduction, Λ_{PS}, of a $\mathsf{PS}, \mathsf{WM}_0$ to $\mathcal{TR}^{PAD}$ is a $\mathcal{TR}^{PAD}$ specification $(\mathcal{E}, \mathbf{P}, \mathcal{S})$ composed of the following PADs ($\mathcal{E}$), rules for defined fluents ($\mathbf{P}$) and premises ($\mathcal{S}$). From now on we assume that the ontology $\mathcal{T}$ is *Datalog-rewritable.*

1. **Ontology $\mathcal{T}$: P** contains all the rules from the Datalog rendering of $\mathcal{T}$.
2. **Initial Database:** The premises below characterize the content of the initial working memory WM_0. For each atomic literal $p(t_1, \ldots, t_n)$ in WM_0

$$\mathbf{d}_0 \rhd p(t_1, \ldots, t_n) \in \mathcal{S} \qquad \mathbf{d}_0 \rhd \mathsf{inertial}(p(t_1, \ldots, t_n)) \in \mathcal{S}$$

   We could have written this as $\mathsf{inertial}(p, t_1, \ldots, t_n)$ to avoid the appearance of being second order or that the use of function symbols here is essential.
3. **Frame Axioms:** The following frame axioms encode the laws of inertia. They also take care of the actual "removal" of $\mathcal{L}_{\mathsf{PS}}$ atoms from the working memory, and the cleaning of the used assignments. Let p be a fluent predicate and α an action such that either $\alpha = \mathit{add_used}$, or $\alpha = \mathit{clean_used}$ or α involves assertion or retraction of an atom with predicate symbol q, where $p \neq q$. For any such pair (p, α), $\mathcal{E}$ has the following PAD except when $\alpha = \mathit{clean_used}$ and $p = \mathsf{used}$:

$$(\mathsf{inertial}(p(\boldsymbol{X})) \wedge p(\boldsymbol{X})) \otimes \alpha(\boldsymbol{Y}) \otimes \mathbf{not}\ \mathsf{inconsistent} \rightarrow \\ \alpha(\boldsymbol{Y}) \otimes (p(\boldsymbol{X}) \wedge \mathsf{inertial}(p(\boldsymbol{X})))$$

   Therefore, the application of the action *clean_used* leads to a state that does not contain any atom with predicate used. If α inserts an atom with predicate symbol q, the frame axiom needs an additional condition of the form $\boldsymbol{X} \neq \boldsymbol{Y}$ on the left side of the above PAD.
4. **Actions:** The following rules encode ***assert*** and ***retract*** in $\mathcal{TR}^{PAD}$:
   - *Insert:* For each predicate $p \in \mathcal{L}_{\mathsf{PS}}$ (whether in $\mathcal{P}_{DL}$ or $\mathcal{P}_{PS}$):

$$\left\{ \begin{array}{r} \mathit{Ins_}p(t_1, \ldots, t_n) \rightarrow \mathit{Ins_}p(t_1, \ldots, t_n) \otimes \\ (p(t_1, \ldots, t_n) \wedge \mathsf{inertial}(p(t_1, \ldots, t_n))) \end{array} \right\} \in \mathcal{E}$$

   - *Retract:* For each predicate $p \in \mathcal{P}_{DL}$,

$$\left\{ \begin{array}{r} \mathit{del_}p(t_1, \ldots, t_n) \rightarrow \mathit{del_}p(t_1, \ldots, t_n) \otimes \\ (\mathbf{neg}\, p(t_1, \ldots, t_n) \wedge \mathsf{inertial}(\mathbf{neg}\, p(t_1, \ldots, t_n))) \end{array} \right\} \in \mathcal{E}$$

oarieli@mta.ac.il

Recall that the effect of the *pda* $\mathit{del_p}$ for PS atoms is given by the interaction with the frame axioms. For instance, if applying $\mathit{del}_{dnaT}(pcr)$ in $\mathbf{d}_1$ results in a state $\mathbf{d}_2$, it holds that $\mathbf{d}_2$ is equal to $\mathbf{d}_1$ except for $\mathsf{dnaT}(pcr)$, which is not carried to $\mathbf{d}_2$ by the frame axioms. This is equivalent to remove $\mathsf{dnaT}(pcr)$ from $\mathbf{d}_2$.

5. **Production rules:** The following rules encode the production rules.
   - For each `IF-THEN`-rule of the form "r: `Forall` $\boldsymbol{x}$: `if` $\phi_r(\boldsymbol{x})$ `then` $\psi_r(\boldsymbol{x})$"

$$r \leftarrow \mathsf{fireable_r}(\boldsymbol{X}) \otimes \widehat{\psi}_r(\boldsymbol{X}) \in \mathbf{P}$$

   - For each `FOR`-rule of the form "r: `For` $\boldsymbol{x}$: $\phi_r(\boldsymbol{x})$ `do` $\psi_r(\boldsymbol{x})$"

$$\left.\begin{array}{l} r \leftarrow \mathsf{fireable_r}(\boldsymbol{X}) \otimes \widehat{\psi}_i(\boldsymbol{X}) \otimes \mathit{add_used}(\boldsymbol{X}) \otimes \mathit{loop_r} \\ \mathit{loop_r} \leftarrow r \\ \mathit{loop_r} \leftarrow (\mathbf{not}\, \exists \boldsymbol{X} : \mathsf{fireable_r}(\boldsymbol{X})) \otimes \mathit{clean_used} \end{array}\right\} \in \mathbf{P}$$

   where $\mathbf{not}\, \exists \boldsymbol{X} : \mathsf{fireable_r}(\boldsymbol{X})$ above is a shorthand for $\mathbf{not}\, \mathsf{p}'$ such that p' is a new predicate defined as $\mathsf{p}' \leftarrow \mathsf{fireable_r}(\boldsymbol{X})$.

6. **Auxiliary Actions and Premises**
   - *Run-Premises*: Then for each pda α and a sequence ξ of actions Ins, del, $\mathit{add_used}$, or $\mathit{clean_used}$, the set of premises $\mathcal{S}$ contains the following run-premise:

$$\mathbf{d}_\xi \overset{a}{\rightsquigarrow} \mathbf{d}_{\xi,a}$$

   For example, $\mathbf{d}_{0,\mathit{Ins_p}(c)} \overset{\mathit{Ins}_q(d)}{\rightsquigarrow} \mathbf{d}_{0,\mathit{Ins_p}(c),\mathit{Ins}_q(d)}$.
   - *Inconsistency*: For each predicate $p \in \mathcal{L}_{\mathsf{PS}}$, $\mathbf{P}$ contains a rule of the form:

$$\mathsf{inconsistent} \leftarrow p(\boldsymbol{X}),\ \mathbf{neg}\, p(\boldsymbol{X})$$

   - *Adding* used *assignments:*

$$\left\{ \mathit{add_used}(\boldsymbol{Y}) \rightarrow \mathit{add_used}(\boldsymbol{Y}) \otimes \mathsf{used}(\boldsymbol{X}) \wedge \mathit{inertial}(\mathsf{used}(\boldsymbol{X})) \right\} \in \mathcal{E}$$

   - *Fireability.* The following rules are in $\mathbf{P}$:

$$\begin{array}{ll} \text{If } r \text{ is an} & \left\{ \begin{array}{l} \mathsf{fireable_r}(\boldsymbol{X}) \leftarrow \widehat{\phi}_r(\boldsymbol{X}) \wedge \\ \qquad (\Diamond \widehat{\psi}_r(\boldsymbol{X}) \otimes \mathbf{not}\, \mathsf{inconsistent}) \end{array} \right. \\ \text{`IF-THEN' rule} & \\ \text{If } r \text{ is a} & \left\{ \begin{array}{l} \mathsf{fireable_r}(\boldsymbol{X}) \leftarrow \widehat{\phi}_r(\boldsymbol{X}) \wedge \mathbf{not}\, \mathsf{used}(\boldsymbol{X}) \wedge \\ \qquad (\Diamond \widehat{\psi}_r(\boldsymbol{X}) \otimes \mathbf{not}\, \mathsf{inconsistent}) \end{array} \right. \\ \text{`FOR' rule} & \end{array}$$

   - *Random choice of action*: For each rule label $r_i \in L$

$$\mathit{act} \leftarrow r_i \in \mathbf{P}$$

To run k rules of the production system we use the transaction:[7]

$$?\text{-}\ \underbrace{\mathit{act} \otimes \cdots \otimes \mathit{act}}_{k}$$

[7] Here we could also use recursion to represent runs of arbitrary length.

oarieli@mta.ac.il

Theorem 2 (Soundness [24]). *Let* $(\mathcal{E}, \mathbf{P}, \mathcal{S})$ *be the* $\mathcal{TR}^{PAD}$ *embedding of a PS configuration. Suppose*

$$\mathcal{E}, \mathbf{P}, \mathcal{S}, \boldsymbol{d}_0 \ldots \boldsymbol{d}_k \models \underbrace{\mathit{act} \otimes \cdots \otimes \mathit{act}}_{m}$$

Then there are working memories $\mathsf{WM}_1 \ldots \mathsf{WM}_m$*, and rules* $r_1 \ldots r_m$ *such that*

$$\begin{array}{c} \mathsf{WM}_0 \overset{r_1}{\hookrightarrow} \mathsf{WM}_1 \\ \vdots \\ \mathsf{WM}_{m-1} \overset{r_m}{\hookrightarrow} \mathsf{WM}_m \end{array}$$

Example 7. In the previous examples, we worked with a simplified version of the frame axioms. Let us now show the complete definition of one of them:

$$\begin{aligned} \mathit{inertial}(\mathsf{dnaT}(T)) \wedge \mathsf{dnaT}(T) \otimes \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathbf{not}\ \mathsf{inconsistent} \rightarrow \\ \mathit{insert}_{\mathsf{takeT}}(P,T) \otimes \mathsf{dnaT}(T) \wedge \mathit{inertial}(\mathsf{dnaT}(T)) \end{aligned}$$

Now we are ready to define the `FOR` rule r_2 and the defined fluent $\mathsf{fireable_}r_2$

$$\begin{aligned} r_2 &\leftarrow \mathsf{fireable_}r_2(P,T) \otimes \mathit{del}_{\mathbf{neg}\,\mathsf{healthy}}(P) \otimes \mathit{add_used}(P,T) \otimes \mathit{loop_r} \\ \mathit{loop_r} &\leftarrow r_2 \\ \mathit{loop_r} &\leftarrow (\mathbf{not}\ \exists P,T\colon \mathsf{fireable_}r_2(P,T)) \otimes \mathit{clean_used} \end{aligned}$$

$$\begin{aligned} \mathsf{fireable_}r_2(P,T) \leftarrow \mathsf{takesT}(P,T) \wedge \mathsf{dnaT}(T) \wedge \mathbf{not}\ \mathsf{used}(\boldsymbol{X})) \wedge \\ (\Diamond \mathit{del}_{\mathbf{neg}\,\mathsf{healthy}}(P) \otimes \mathbf{not}\ \mathsf{inconsistent}) \end{aligned}$$

Now we come back to our query, and check: $\texttt{?-}_{\mathbf{d}_0}\mathsf{flu}(\mathrm{Laura}) \otimes \mathit{r_1}$.

In this case we would detect an inconsistency produced by *r_1*. By observing the construction of the well founded model, we can track back the conflict. □

Recall that beside detecting inconsistencies, $\mathcal{TR}^{PAD}$ allow to execute inconsistency free rules and obtain runs as in the original production system.

7 Conclusions

In this paper we proposed a new semantics for the combination of production systems with *arbitrary DL* ontologies. Unlike previous approaches [26,13,10,4,21,31], the semantics presented here supports extensions, like the *FOR*-loops or *while*-loops, that are not included in RIF-PRD, but are found in commercial production systems such as IBM's *JRules* [19]. In addition, our approach can handle inconsistencies produced by the interaction of rule actions and the ontology.

We also defined a sound embedding of such semantics, restricted to rule-based *DL* ontologies, into Transaction Logic with partial action definitions ($\mathcal{TR}^{PAD}$). This reduction gives a declarative semantics to the combination, and is considerably simpler and compact that other approaches, including [26,21,31,10,20].

To model production systems in $\mathcal{TR}^{PAD}$, we extended $\mathcal{TR}^{PAD}$ with default negation and defined the well-founded semantics [29] for it. It is worth noting

oarieli@mta.ac.il

that this $\mathcal{TR}^{PAD}$ embedding can be used as an implementation vehicle for the combination of PS and rule-based ontologies.

Acknowledgments. We thank the anonymous reviewers for useful feedback. M. Rezk and M. Kifer were partially supported by the European Commission under the project OntoRule. M. Kifer was also partially supported by the NSF grant 0964196.

References

1. Anicic, D., Fodor, P., Stühmer, R., Stojanovic, N.: An approach for data-driven logic-based complex event processing. In: The 3rd ACM International Conference on Distributed Event-Based Systems, DEBS (2009)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The dl-lite family and relations. Journal of Artificial Intelligence Research (JAIR) 36, 1–69 (2009)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook. Cambridge University Press (2003)
4. Baral, C., Lobo, J.: Characterizing production systems using logic programming and situation calculus (1995), http://www.public.asu.edu/~cbaral/papers/char-prod-systems.ps
5. Bonner, A., Kifer, M.: Transaction logic programming. In: Proc. of International Conference on Logic Programming (ICLP), Budapest, Hungary, pp. 257–282. MIT Press (June 1993)
6. Bonner, A., Kifer, M.: Transaction logic programming (or a logic of declarative and procedural knowledge). Technical Report CSRI-323, University of Toronto (November 1995), http://www.cs.sunysb.edu/~kifer/TechReports/transaction-logic.pdf
7. Bonner, A.J., Kifer, M.: A logic for programming database transactions. In: Chomicki, J., Saake, G. (eds.) Logics for Databases and Information Systems, ch. 5, pp. 117–166. Kluwer Academic Publishers (March 1998)
8. Bonner, A.J., Kifer, M.: Applications of Transaction Logic to Knowledge Representation. In: Gabbay, D.M., Ohlbach, H.J. (eds.) ICTL 1994. LNCS, vol. 827, pp. 67–81. Springer, Heidelberg (1994)
9. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of $DL-Lite$ Knowledge Bases. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 112–128. Springer, Heidelberg (2010)
10. Damásio, C.V., Alferes, J.J., Leite, J.: Declarative Semantics for the Rule Interchange Format Production Rule Dialect. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 798–813. Springer, Heidelberg (2010)
11. Davulcu, H., Kifer, M., Ramakrishnan, C.R., Ramakrishnan, I.V.: Logic based modeling and analysis of workflows. In: PODS, pp. 25–33 (1998)
12. Davulcu, H., Kifer, M., Ramakrishnan, I.V.: Ctr-s: a logic for specifying contracts in semantic web services. In: WWW, pp. 144–153 (2004)
13. de Bruijn, J., Rezk, M.: A Logic Based Approach to the Static Analysis of Production Systems. In: Polleres, A., Swift, T. (eds.) RR 2009. LNCS, vol. 5837, pp. 254–268. Springer, Heidelberg (2009)

oarieli@mta.ac.il

14. de Sainte Marie, C., Hallmark, G., Paschke, A.: Rif production rule dialect (2010), http://www.w3.org/TR/rif-prd/
15. Emerson, E.A.: Temporal and modal logic. In: Handbook of Theoretical Computer Science, pp. 995–1072. Elsevier (1995)
16. Fodor, P., Kifer, M.: Transaction logic with defaults and argumentation theories. In: ICLP (Technical Communications), pp. 162–174 (2011)
17. Heymans, S., Eiter, T., Xiao, G.: Tractable reasoning with dl-programs over datalog-rewritable description logics. In: European Conference on Artificial Intelligence, pp. 35–40 (2010)
18. Horrocks, I.: Ontologies and the semantic web. Commun. ACM 51, 58–67 (2008)
19. I. JRules, http://www.ibm.com/software/integration/business-rule-management/jrules-family/
20. Kowalski, R., Sadri, F.: Integrating Logic Programming and Production Systems in Abductive Logic Programming Agents. In: Polleres, A., Swift, T. (eds.) RR 2009. LNCS, vol. 5837, pp. 1–23. Springer, Heidelberg (2009)
21. Lausen, G., Ludäscher, B., May, W.: On Active Deductive Databases: The Statelog Approach. In: Kifer, M., Voronkov, A., Freitag, B., Decker, H. (eds.) Dagstuhl Seminar 1997, DYNAMICS 1997, and ILPS-WS 1997. LNCS, vol. 1472, pp. 69–106. Springer, Heidelberg (1998)
22. Pearce, D., Wagner, G.: Logic Programming with Strong Negation. In: Eriksson, L.-H., Hallnäs, L., Schroeder-Heister, P. (eds.) ELP 1991. LNCS, vol. 596, pp. 311–326. Springer, Heidelberg (1992)
23. Raschid, L.: A semantics for a class of stratified production system programs. J. Log. Program. 21(1), 31–57 (1994)
24. Rezk, M., Kifer, M.: Formalizing production systems with rule-based ontolgies (2011), http://www.inf.unibz.it/~mrezk/techreportTRPS.pdf
25. Rezk, M., Kifer, M.: Reasoning with Actions in Transaction Logic. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 201–216. Springer, Heidelberg (2011)
26. Rezk, M., Nutt, W.: Combining Production Systems and Ontologies. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 287–293. Springer, Heidelberg (2011)
27. Roman, D., Kifer, M.: Reasoning about the behavior of semantic web services with concurrent transaction logic. In: VLDB, pp. 627–638 (2007)
28. Roman, D., Kifer, M.: Semantic Web Service Choreography: Contracting and Enactment. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 550–566. Springer, Heidelberg (2008)
29. Van Gelder, A., Ross, K., Schlipf, J.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)
30. Winslett, M.: Updating logical databases. Cambridge University Press, New York (1990)
31. Zaniolo, C.: A unified semantics for active and deductive databases. In: Workshop on Rules In Database Systems (RIDS 1993), pp. 271–287. Springer, Heidelberg (1993)

oarieli@mta.ac.il

Count Constraints and the Inverse OLAP Problem: Definition, Complexity and a Step toward Aggregate Data Exchange

Domenico Saccà, Edoardo Serra, and Antonella Guzzo

DEIS, Università della Calabria, 87036 Rende, Italy
sacca@unical.it, {eserra,guzzo}@deis.unical.it

Abstract. A typical problem in database theory is to verify whether there exists a relation (or database) instance satisfying a number of given dependency constraints. This problem has recently received a renewed deal of interest within the context of data exchange, but the issue of handling constraints on aggregate data has not been much investigated so far, notwithstanding the relevance of aggregate operations in exchange systems. This paper introduces count constraints that require the results of given count operations on a relation to be within a certain range. Count constraints are defined by a suitable extension of first order predicate calculus, based on set terms, and they are then used in a new decisional problem, the *Inverse OLAP*: given a star schema, does there exist a relation instance satisfying a set of given count constraints? The new problem turns out to be NEXP complete under various conditions: program complexity, data complexity and combined complexity. Count constraints can be also used into a data exchange system context, where data from the source database are transferred to the target database using aggregate operations.

Keywords: Count Constraints, Star Schema, OLAP, NEXP, Database Integrity Check, Inverse Data Mining, Data Exchange.

1 Introduction

Inverse frequent set mining (IFM) [20,8,9], consists of verifying whether there exists a database D of transactions over a set $\mathcal{I}$ of items satisfying a given set Γ_S of support constraints on some itemsets, that are typically the frequent ones. The original aim of our research was mainly to present a formulation of IFM problem in which support constraints are given in a succinct and flexible way using generic properties of them. The search for a general framework stimulated us to consider the problem under a new perspective: as a special case of constraint satisfaction for an OLAP star schema.

IFM can be formulated as a particular problem of integrity constraint satisfaction on a relational scheme $\mathcal{R}(T, I)$ with two attributes T (the ID of a transaction) and I (the ID of an item) with domain $\mathcal{I}$, stored in a relation $\mathcal{D}_I$. Given a relation r on $\mathcal{R}$, $G = Group(r)By(T)$ divides r into a number of groups, one for each transaction ID. Then, for each $K \subseteq \mathcal{I}$, $G^{\sigma}(K) = G_{(I \supseteq K)}$ filters the groups in G for which K is a subset of the projection on I. We then consider a new type of integrity constraint for $\mathcal{R}$, called

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 352–369, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

count constraint, requiring that, for each relation r and for each $K \subseteq \mathcal{I}$, the number of filtered groups in $G^{\sigma}(K)$ be within a given range — thus enforcing support restrictions.

More in general count constraints prescribe that the results of given count operations on a relation to be within a certain range. To enable an expressive description of count constraints, we extend the logic formalism adopted in the data exchange setting [13,5,14] to handle set terms. We anticipate the flavor of such a formalism by showing how to express the following IFM problem: the two itemsets $i = \{a, b, c\}$ and $j = \{b, c, d\}$) must be present in (i.e. must have a support of) at least 100 and in at most 200 transactions, whereas every other set s of items cannot appear in more than 15 transactions, except for all subsets of i and j that have no limits:

$$\forall s\,(\, s = i \vee s = j \;\rightarrow\; 100 \leq \#(\{x_T : s \subseteq \{x_I :\, \mathcal{R}(x_T, x_I)\}\}) \leq 200\,);$$
$$\forall s\,(\, s \subseteq \{x_I : \mathcal{D}_I(x_I)\} \wedge s \not\subset i \wedge s \not\subset j \rightarrow 0 \leq \#(\{x_T : s \subseteq \{x_I : \mathcal{R}(T, I)\}\}) \leq 10\,).$$

A light version of count constraints are cardinality constraints that have been recently analyzed in [4] with the aim of generating an instance of a database scheme satisfying such constraints. The expressive power of cardinality constraints is easily captured by our formalism. For instance, the constraints that the item a be present in exactly 1000 tuples in any relation r or $\mathcal{R}$ is expressed as $|\pi_{T,I}\sigma_{I='a'}(\mathcal{R})| = 1000$ in the formalism of [4], while we write:

$$true \;\rightarrow 1000 \leq \#(\{x_T :\, \mathcal{R}(x_T, 'a')\}) \leq 1000\,).$$

Cardinality constraints cannot be applied to sets generated by grouping operations, as we have instead done in the formulation of the above IFM problem.

We use count constraints to define a new decisional problem, the *Inverse OLAP*: given a star schema, does there exist a relation instance satisfying a set of given count constraints? A star schema is a relation scheme used in OLAP analisys [10,15] to represent multidimensional data cubes: each *dimension* of a multidimensional data cube stores the values of an attribute and each cell contains an aggregate value, called *measure* (e.g., count), based on aggregation criteria that depend on coordinates of this cell in the data cube.

Inverse OLAP is relevant for the generation of synthetic data cubes having the same characteristics of real-world ones in terms of aggregation patterns. The formulation of this problem shows a strict relationship between classical integrity constraint satisfaction on database schemes and inverse mining problems: both problems provide a succinct description of satisfying instances. In the paper we prove that the new problem is NEXP-complete under various conditions: data complexity (i.e, the number of attributes and the size of constraints are constant) and combined complexity.

The paper is organized as follows. In Section 2 we present some useful background to our research. In Section 3 we introduce the logic language for describing count constraints and illustrate their usage in a motivating example in Section 4. In Section 5 we present the formulation of the Inverse OLAP and prove that it is in NEXP under various conditions. In Section 6 we illustrate how count constraints can be used into a data exchange system context, where data from the source database are transferred to the target database using aggregate operations. Finally in Section 7 we draw the conclusion and argue that, following the approach used in a recent paper for finding efficient solutions to IFM [16], Inverse OLAP could be solved in a reasonable amount of time also for a large input size despite its high complexity.

oarieli@mta.ac.il

2 Related Work

The issues of count constraints and inverse OLAP are rather novel and, at the best of our knowledge, there is not much direct related work. Classical problems of deciding the existence of a database satisfying a given set of integrity constraints refer to a relational scheme and consider classes of constraints which admit a finite *chase* [6], i.e., a finite "canonical" database, witnessing the satisfiability of the constraints at hand. In particular, most of such classical studies in database theory focused on inclusion dependencies and functional dependencies [29,24,11].

OLAP analysis is characterized by multidimensional data cubes that enable manipulating and analyzing data stored in a source database from multiple perspectives in a fast way [10,15]. A data cube is defined by designing a relational database schema as a multidimensional view on a source database, typically a *star schema* representing a fact table, whose attributes are measures and dimensions are stored. Measures are values computed on the basis of some aggregation operations. Dimensions are properties that can be structured at various levels of abstraction – for instance, the dimension `store` of a `sell` fact may refer to another dimension `region` so that it will be eventually possible to group not only all the sales of a store but also those of an entire region. After the definition of the star schema, the data cube is built by performing a number of Projection/Selection/GroupBy queries over the source database. In case of two dimensions A and B for which the functional dependency $A \rightarrow B$ holds (*layered dimensions*), the dimension B can be removed from the fact table and stored into an additional ad-hoc (dimension) table $D(A, B)$ having A as key, thus obtaining a so called *snowflake schema*; in this case an inclusion dependency is added which enforces that all values of the dimension A in the fact table be contained in the corresponding column of the dimension table.

We observe that a version of count constraints called cardinality constraints have recently received a renewed interest. A declarative format to express them has been recently introduced in [4] with the aim of formulating more general characteristics in the process of data generation, that is perfomed by tools such as QAGen [7] and its extension MyBenchmark [19]. Cardinality constraints have been also introduced within the formalism of logic programs by [26]. It is interesting to point out that the NEXP-complete results on the complexity of cardinality constraints have been detected both in [4] and in [26].

Inverse OLAP has an important connection with another inverse problem, called inverse frequent itemset mining. Let $\mathcal{I}$ be a finite domain of elements, also called *items*. Any subset $I \subseteq \mathcal{I}$ is an *itemset* over $\mathcal{I}$, usually also called *transaction*. A database $\mathcal{D}$ over $\mathcal{I}$ is a bag of itemsets, thus transactions may occur duplicated in $\mathcal{D}$. For each itemset $I \in D$, there exists two important measures: (i) the *number of duplicates* of I, and (ii) the *support* of I, that is the sum of all number of duplicates of itemsets in $\mathcal{D}$ containing I. A popular mining task over transaction databases is to single out the set of the *frequent itemsets*, i.e., all the subsets of $\mathcal{I}$ (called *itemsets*) which are contained in a significant fraction (user-specified as a minimum support threshold) of the given transactions [2].

The *inverse frequent itemset mining* (short: `IFM`) problem consists of computing a transactional database (or answer that there is no transactional database) on which every

oarieli@mta.ac.il

itemset in a collection provided as input is frequent [20,8,9]. This problem attracted attention in the recent years, due to its potential usage in privacy preserving data mining scenarios [3,28] and in generating synthetic databases having the same characteristics of real-world ones (in terms of the frequency of some relevant itemsets) [23].

As we shall illustrate later in the paper, Inverse OLAP may have strict relationships with the data exchange problem, first defined in [13,5,14]. This is the problem of migrating a data instance from a source schema to a target schema such that the materialized data on the target schema satisfies the integrity constraints specified by it. Data exchange is different than data integration [18] because the data is indeed materialized at the target schema, which is not always the case for data integration settings. The mapping of the data from the source to the target schema is given by source- to-target TGDs (Tuple Generating Dependencies). Additionally, the target schema specifies target constraints in form of EGDs (Equality Generating Dependencies) and TGDs, which the imported data must satisfy. Moreover, it was proved in that, the answer of a conjunctive query on a universal solution, is contained in all answers of the same query on each mapping solution. Aggregate queries in the data exchange setting has been investigated in [1].

3 Count Constraints

Let $U = (A_1, \ldots, A_n)$ be a list of n distinct attributes on the domains $D_1, \ldots, D_n$ with given cardinalities $d_1, \ldots, d_n$. We denote $\cup_{i=1,n} D_i$ by $\mathbf{D}$ and $\sum_{i=1,n} d_i$ by $\mathbf{d}$. Given a sublist $L = \{A_{i_1}, \ldots, A_{i_k}\}$ of $k, 0 \leq k \leq n$,distinct attributes in U, a *tuple* $a = [a_1, \ldots, a_k]$ on L is a mapping from $A_{i_1} \times \cdots \times A_{i_k}$ to $D_{i_1}, \ldots, D_{i_k}$. We denote the set of all tuples on L by D^L; a subset (not necessarily proper) of D^L is called a *table* on L. If $k = 0$ then D^L only contains the empty tuple. In addition, given $k > 0$, $[a_1, \ldots, a_k]$ with $a_1, \ldots, a_k \in \mathbf{D}$ will be called an *untyped tuple* and we shall write that $[a_1, \ldots, a_k] \in \mathbf{D}^k$.

A *relation scheme* is a pair consisting of a relation name and a list of attributes. In this paper we shall only deal with exactly one relation scheme containing all attributes in U, say $\mathcal{R}(A_1, \ldots, A_n)$. We also assume that the domains of the attributes $D_1, \ldots, D_n$ are stored in suitable tables of mono-attribute relation schemes — we denote their relation schemes by $\mathcal{D}_1(A_1), \ldots, \mathcal{D}_n(A_n)$. The size of the description of $\mathcal{R}$ and the involved domains is then $\mathbf{d}$ and a *relation* on $\mathcal{R}$ (also called an *instance* of $\mathcal{R}$) is any table on U. $\mathcal{R}$ represents a typical *star schema* whose dimensions are stored into a unique *fact table*, that is a relation on $\mathcal{R}$ - as we are only interested to count aggregation, we omit to include measures. Some of the dimensions could be organized in layers by means of Functional Dependencies (FDs) — for instance the FDs $A \rightarrow B$ and $B \rightarrow C$ state that the values of dimension A are grouped at a first level B and at a second level C. In correspondence of FDs we may have additional domain relations describing hierarchies among two dimensions, e.g., $\mathcal{D}_{A,B}$ and $D_{B,C}$.

We next introduce an extension of first order predicate calculus to define count constraints on the instances of $\mathcal{R}$. The predicate symbols only are $\mathcal{R}$, the domain relation schemes $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and possible dimension hierarchy domains. The *constants* of the language are the domain values (*domain constants*) and all (non-negative) integers.

oarieli@mta.ac.il

Besides to domain constants, the Herbrand universe includes constant set terms defined as follows: given any sublist S of U, a constant set term is a set of tuples (i.e., a table) over S. To avoid to have a super-exponential number of set terms in the universe, we fix a maximum size $\hat{k}$ for the arity of S, thus $\hat{k}$ is a (typically small) constant given as part of the input of our problem — this is not an actual limitation in practical applications for which the value of $\hat{k}$ is often 1. Let $\mathbf{H}_{\hat{k}}$ denote the Herbrand universe; furthermore, given any integer $p \geq 0$, $\mathbf{H}_{\hat{k}}^{p}$ denotes the set of all extended p-tuples on the Herbrand universe – a tuple is extended as an element of it can also be a constant set term.

Given the attributes A and B with domains $\{a_1, a_2, a_3\}$ and $\{b_1, b_2\}$ respectively, examples of constant set terms on $\{A, B\}$ are $\{[a_1, b_1], [a_2, b_1], [a_3, b_2]\}$ and $\{[a_2, b_1]\}$, while $\{[a_1], [a_3]\}$ and $\{[a_2]\}$ are constant set terms on $\{A\}$. There is an interpreted function symbol $count$ (denoted by $\#$) that can be applied to a set term T to return the number of tuples in T (i.e., the cardinality of the table represented by T).

Our language is equipped with a countable number of variables and makes use of the following types of terms:

- *simple term*: it is either a domain constant or a variable;
- *set term*: it is either a constant set term or a *formula term*, defined as $\{x_1, \ldots, x_s : \alpha\}$, where $x_1, \ldots, x_s$ are variables, $s \leq k$ and α is a *count formula* (defined next), in which $x_1, \ldots, x_s$ occur as free variables (similar notation for set terms and aggregate predicates has been used in the dlv system [12]);
- *integer term*: it can be either an integer or a function term $\#(T)$, where T is either a variable or a set term.

An *atom* can have one of the following formats:

- $\mathcal{R}(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are simple terms (*relation predicate*);
- $\mathcal{D}_i(t)$, $\forall 1 \leq i \leq n$, where t is a simple term, and for each dimension hierarchy domain on a pair of attributes A_i and A_j, $\mathcal{D}_{A_i,A_j}(t_1, t_2)$, where t_1 and t_2 are simple terms (*domain predicates*);
- *comparison predicates* of the following types:
  - $t_1 = t_2$ and $t_1 \neq t_2$, where t_1 and t_2 are terms (*equality predicate* and *disequality predicate*, respectively);
  - $t_1 < t_2$ and $t_1 \leq t_2$, where t_1 and t_2 are either variables or integer terms (*less predicates*) — we shall use $t_1 \leq t_2 \leq t_3$ as a shorthand for $t_1 \leq t_2 \wedge t_2 \leq t_3$;
  - $t_1 \subseteq t_2$, $t_1 \subset t_2$, $t_1 \not\subseteq t_2$ and $t_1 \not\subset t_2$, where t_1 and t_2 are variables or set terms (*subset predicate*, *strict subset predicate*, *non-subset predicate* and *strict non-subset predicate*, respectively).

A *count constraint* C is a formula of type:

$$\forall \mathbf{X} \; (\; \alpha \; \rightarrow \; \beta_{min} \leq \#(\{ \; \mathbf{Y} : \gamma \; \}) \leq \beta_{max} \;)$$

where:

- $\mathbf{X}$ and $\mathbf{Y}$ are disjunct lists of variables - $\mathbf{X}$ can be empty;
- β_{min} and β_{max} are integers, but β_{max} can be set to ∞ to denote that the right hand side count predicate simply reduces to $\beta_{min} \leq \#(\{\mathbf{Y} : \alpha\})$;

oarieli@mta.ac.il

- α is a (possibly empty) conjunction of domain and comparison predicates, whose variables are in $\mathbf{X}$ — a possible non-constant set term occurring in α must be of the form $\{\mathbf{W} : \delta\}$, where δ is a conjunction of domain predicates and the variables in $\mathbf{W}$ are distinct from the ones in $\mathbf{X}$ and all occur in δ;
- every variable $x \in \mathbf{X}$ must be *bound* in α, that is:
  - x occurs in a domain predicate, or
  - x occurs in a comparison predicate $x = t$, $t = x$, $x \subset t$ or $x \subseteq t$ such that t is a constant or a bounded variable or a set term $\{\mathbf{W} : \delta\}$ in which all the variables in δ are bound;
- γ can be either
  1. $\forall \mathbf{Z}\ \mathcal{R}(t_1, \ldots, t_n)$, where $\mathbf{Z}$ is a possibly empty list of variables, distinct from the ones in $\mathbf{X}$ and $\mathbf{Y}$, $t_1, \ldots, t_n$ are simple terms, all variables in $\mathbf{Y} \cup \mathbf{Z}$ occur as terms in $\mathcal{R}$ and all variables in $\mathcal{R}$ are in $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$ (*tuple count constraint*), or
  2. $t * \{\ \mathbf{Z_1} : \forall \mathbf{Z_2}\ \mathcal{R}(t_1, \ldots, t_n)\ \}$, where $*$ can be $=$ or $\subset$ or $\subseteq$, t is either a constant set term or a variable in $\mathbf{X}$, $\mathbf{Z_1}$ and $\mathbf{Z_2}$ are disjunct lists of variables distinct from the ones in $\mathbf{X}$ and $\mathbf{Y}$ ($\mathbf{Z_2}$ can be empty), $t_1, \ldots, t_n$ are simple terms, all variables in $\mathbf{Y} \cup \mathbf{Z_1} \cup \mathbf{Z_2}$ occur as terms in $\mathcal{R}$ and all variables in $\mathcal{R}$ are in $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z_1} \cup \mathbf{Z_2}$ (*group count constraint*); moreover, if $*$ is $=$ the constraint is called (*duplicate count constraint*) or (*support count constraint*) .

Given a relation scheme $\mathcal{R}$ together with its domain relations and a relation r on $\mathcal{R}$, a constraint C as above is evaluated on r by a function $eval$ as follows – note that $eval$ return true or false if applied to a predicate and a constant (simple or set) term if applied to a term:

- a relation (resp., domain) predicate $\theta[\mathbf{X}/\mathbf{x}]$, for which all variables $\mathbf{X}$ occurring in θ are replaced by constants $\mathbf{x}$, is true if the corresponding tuple is in r (resp., the domain relation) and false otherwise;
- a comparison predicate $t_1[\mathbf{X_1}/\mathbf{x_1}]\ *\ t_2[\mathbf{X_2}/\mathbf{x_2}]$ is evaluated to true if $eval(t_1[\mathbf{X_1}/\mathbf{x_1}]) * eval(t_2[\mathbf{X_2}/\mathbf{x_2}])$[1] or to false otherwise;
- for each term $t[\mathbf{X}/\mathbf{x}]$, $eval$ is defined as follows:
  - if t is a simple term then $eval(t[\mathbf{X}/\mathbf{x}]) = t[\mathbf{X}/\mathbf{x}]$;
  - if $t[\mathbf{X}/\mathbf{x}] = \#(t'[\mathbf{X}/\mathbf{x}])$ and $eval(t'[\mathbf{X}/\mathbf{x}])$ is a set s then $eval(t[\mathbf{X}/\mathbf{x}]) = |s|$;
  - if $t[\mathbf{X}/\mathbf{x}] = \{\mathbf{Y} : \forall \mathbf{Z}\ \theta\}$, where θ is a relation or domain predicate, $|\mathbf{Y}| = p > 0$ and $|\mathbf{Z}| = q \geq 0$, and all the variables in θ are in $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$, then $eval(t[\mathbf{X}/\mathbf{x}]) = \bigcup_{\mathbf{z} \in \mathbf{D}^q} \{\mathbf{y} \in \mathbf{D}^p \mid \theta[\mathbf{X}/\mathbf{x}, \mathbf{Y}/\mathbf{y}, \mathbf{Z}/\mathbf{z}]$ is evaluated to true$\}$;
  - if $t[\mathbf{X}/\mathbf{x}] = \{\ \mathbf{Y} : t'[\mathbf{X}/\mathbf{x}] * \{\ \mathbf{Z_1} : \forall \mathbf{Z_2}\ \theta[\mathbf{Y}, \mathbf{Z_1}, \mathbf{Z_2}]\ \}\ \}$, where θ is a relation predicate, t' is a term, $|\mathbf{Y}| = p > 0$, $|\mathbf{Z_1}| > 0$ and $|\mathbf{Z_2}| \geq 0$, and the variables in θ are in $\mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z_1} \cup \mathbf{Z_2}$, then $eval(t[\mathbf{X}/\mathbf{x}]) = \{\mathbf{y} \in \mathbf{D}^p \mid eval(t'[\mathbf{X}/\mathbf{x}]) * eval(\{\ \mathbf{Z_1} : \forall \mathbf{Z_2}\ \theta[\mathbf{X}/\mathbf{x}, \mathbf{Y}/\mathbf{y}]\ \})\ \}$;
- the conjunction of two predicates $\alpha_1[\mathbf{X}/\mathbf{x}] \wedge \alpha_2[\mathbf{X}/\mathbf{x}]$ is evaluated to true if so are both predicates or to false otherwise;

[1] The semantics of the various comparison operators * on evaluated terms is the classical interpretation of comparisons between constants.

oarieli@mta.ac.il

– $\forall \mathbf{X}\ (\ \alpha\ \rightarrow\ \beta_{min}\ \leq\ \#(\{\ \mathbf{Y}\ :\ \gamma\ \})\ \leq\ \beta_{max}\ )$, say with $|\mathbf{X}| = p \geq 0$, is evaluated (i) to true if for each $\mathbf{x} \in \mathbf{H}^p_{\hat{k}}$, either $\alpha[\mathbf{X}/\mathbf{x}]$ is evaluated to false or $(\beta_{min} \leq \#(\{\ \mathbf{Y}\ :\ \gamma\ \}) \leq \beta_{max})[\mathbf{X}/\mathbf{x}]$ is evaluated to true, and (ii) to false otherwise — $\mathbf{x}$ is any extended p-tuple on the Herbrand universe, i.e., it can be a set term.

Observe that in the above definitions we have used untyped tuples to simplify notation - this simplification is not a restriction as the correct domain type is eventually enforced by the occurrence of tuple values in the appropriate domain in the relation $\mathcal{R}$. Also note that, on the basis of its semantics, $\{\ \mathbf{Z_1}\ :\ \forall \mathbf{Z_2}\ \mathcal{R}(t_1, \ldots, t_n)\ \}$, is actually a shorthand for $\{\ \mathbf{Z_1}\ :\ \wedge_{Z \in \mathbf{Z_1}} \mathcal{D}(Z) \wedge \forall \mathbf{Z_2}\ \mathcal{R}(t_1, \ldots, t_n)\ \}$.

We finally remark that the actual range of a variable X in a count constraint is restricted by the bound conditions we have introduced — we can then say that computation is in general safe. A potential range explosion could arise in a comparison predicate $X \subset t$ or $X \subseteq t$, but this risk is mitigated for the term t eventually unifies with a constant set of tuples and by the restriction on the maximum set arity $\hat{k}$.

A relation r satisfies a count constraint C (and we write $C \models r$) if the evaluation of C on r is equal to true. Accordingly, r satisfies a set of count constraints $\mathbf{C}$ (and we write $\mathbf{C} \models r$) if for each $C \in \mathbf{C}$, $C \models r$.

As stated in the proposition below, checking count constraint satisfaction may require exponential time. We omit to include the proof for space reasons. This proof as well as the ones of many other results of this paper can be found in [25].

Proposition 1. *Given a relation scheme $R(A_1, \ldots, A_n)$ with $n > 0$, the domains $D_1, \ldots, D_n$ with possibly a fixed additional number of hierarchy domains, a maximal set arity $\hat{k}$, a count constraint C and a relation r on R, deciding $C \models r$ is in* EXP.

Particularly interesting are constraints for which satisfaction can be tested in polynomial time. We say that count constraints with this property are *well structured*.

4 A Motivating Example

We refer to a classical example of point-of-sales transaction star schema. The attributes of U are: T (Transaction), I (Item), B (Brand), S (Store), A (Area) — their (finite) domains can be suitably defined. We are also given the following functional dependencies (FDs): $T \rightarrow S$, $S \rightarrow A$. It turns out that $\{T, I, B\}$ is the relation key. The domains of the attributes are denoted by $\mathcal{D}_T$, $\mathcal{D}_I$ and so on. We next present a number of meaningful count constraints that clarify their usage:

(i): Enforcing FDs and relation key
For instance the FD $T \rightarrow S$ can be expressed as follows:

$$\forall x_T(\ \mathcal{D}_T(x_T)\ \rightarrow\ 0 \leq \#(\{x_S : \forall x_I, x_B, x_A\ \mathcal{R}(x_T, x_I, x_B, x_S, x_A)\}) \leq 1\ )$$

The relation key $\{T, I, B\}$ can be enforced as:

$$\forall x_T, x_I, x_B(\mathcal{D}_T(x_T) \wedge \mathcal{D}_I(x_I) \wedge \mathcal{D}_B(x_B) \rightarrow$$
$$0 \leq \#(\{x_S, x_A : \mathcal{R}(x_T, x_I, x_B, x_S, x_A))\}) \leq 1)$$

oarieli@mta.ac.il

(ii): Enforcing the overall number of tuples

There must be between 50000 and 100000 tuples in any instance of R:

$$true \rightarrow 50000 \leq \#(\{x_T, x_I, x_B, x_S, x_A : \mathcal{R}(x_T, x_I, x_B, x_S, x_A)\}) \leq 100000$$

(iii): Enforcing the total number of transactions in an area

There must be between 1000 and 2000 transactions in every region, except in "Cal" ("Calabria" not "California")for which the upper bound is increased to 9000:

$$true \rightarrow 1000 \leq \#(\{x_T : \forall x_I, x_B, x_S\ \mathcal{R}(x_T, x_I, x_B, x_S, "Cal")\}) \leq 9000;$$
$$\forall x_A(\mathcal{D}_A(x_A) \wedge x_A \neq "Cal" \rightarrow$$
$$1000 \leq \#(\{x_T : \forall x_I, x_B, x_S \mathcal{R}(x_T, x_I, x_B, x_S, x_A)\}) \leq 2000).$$

If we wish to enforce the above transaction constraint in every store of an area, we can use the dimension hierarchy domain $\mathcal{D}_{S,A}$:

$$\forall x_S(\mathcal{D}_{x_S,x_A}(x_S, "Cal") \rightarrow 1000 \leq \#(\{x_T : \forall x_I, x_B\ \mathcal{R}(x_T, x_I, x_B, x_S, "Cal")\}) \leq 9000);$$
$$\forall x_A, x_S\ (\ \mathcal{D}_{x_S,x_A}(x_S, x_A) \wedge x_A \neq "Cal" \rightarrow$$
$$1000 \leq \#(\{x_T : \forall x_I, x_B\ \mathcal{R}(x_T, x_I, x_B, x_S, x_A)\}) \leq 2000\ ).$$

(iv): 1-arity group count constraints

Both the set of items $i = \{[a], [b], [c]\}$ and $j = \{[b], [c], [d]\}$) must be present in at least 100 and in at most 200 transactions, whereas every other set s of items cannot appear in more than 15 transactions if s contains more than 10 elements or 20 otherwise, except for all subsets of i and j that have no limits (for space reasons we below write i and j to represent the two constant set terms):

$$\forall s(s = i \vee s = j \rightarrow 100 \leq \#(\{x_T : s \subseteq \{x_I : \forall x_B, x_S, x_A \mathcal{R}(x_T, x_I, x_B, x_S, x_A)\}\}) \leq 200);$$
$$\forall s\ (\ s \subseteq \{x_I : \mathcal{D}_I(x_I)\} \wedge s \not\subset i \wedge s \not\subset j \wedge \#(s) \leq 10 \rightarrow$$
$$0 \leq \#(\{x_T : s \subseteq \{x_I : \forall x_B, x_S, x_A\ \mathcal{R}(x_T, x_I, x_B, x_S, x_A)\}\}) \leq 20\ );$$
$$\forall s\ (\ s \subseteq \{x_I : \mathcal{D}_I(x_I)\} \wedge s \not\subset i \wedge s \not\subset j \wedge \#(s) > 10 \rightarrow$$
$$0 \leq \#(\{x_T : s \subseteq \{x_I : \forall x_B, x_S, x_A\ \mathcal{R}(x_T, x_I, B, x_S, x_A)\}\}) \leq 15\ ).$$

Note that the first of the above constraints has a disjunction in the left hand side: it is only a shorthand to represent two constraints having the same right hand side.

Observe that constraint define an instance of an `IFM` problem for which, in addition to fixing support constraints for a number of pre-defined itemsets (typically the frequent ones, in this case i and j), there are generic support constraints for all other itemsets (the unfrequent ones).

If we now replace the operator $\subseteq$ with $=$ in the count terms on the right hand sides of the above constraints, then the count bounds hold only when an itemset indicated by $\hat{I}$ contain exactly all items of a transaction. So, in terms of the mentioned `IFM` problem, this revised formulation introduces duplicate constraints.

(iv): 2-arity group count constraints

There must be at least 100 and at most 200 transactions containing an item "sm" (smartphone) of the brand "nd" (ndrangtung), whereas the same set of pairs of item and brand are sold together in at most 10 transactions, except for the ones containing the pair ("sm","nd") for which the limit is 50 (for space reasons we write t to represent the singleton constant set term $\{["sm","nd"]\}$):

oarieli@mta.ac.il

$$true \rightarrow 100 \leq \#(\{x_T : t \subseteq \{x_I, x_B : \forall x_S, x_A\, \mathcal{R}(x_T, x_I, x_B, x_S, x_A))\}\}) \leq 200;$$
$$\forall s\ (\ s \subseteq \{x_I, x_B : \mathcal{D}_I(x_I) \wedge \mathcal{D}_B(x_B)\} \wedge t \subset s\ \rightarrow$$
$$0 \leq \#(\{x_T : s \subseteq \{x_I, x_B : \forall x_S, x_A\, \mathcal{R}(x_T, x_I, x_B, x_S, x_A))\}\}) \leq 50\ );$$
$$\forall s\ (\ s \subseteq \{x_I, x_B : \mathcal{D}_I(x_I) \wedge \mathcal{D}_B(x_B)\} \wedge t \not\subset s\ \rightarrow$$
$$0 \leq \#(\{x_T : s \subseteq \{x_I, x_B : \forall x_S, x_A\, \mathcal{R}(x_T, x_I, x_B, x_S, x_A))\}\}) \leq 10\ ).$$

(Recall that $\subset$ denotes strict subset relationship.) The above constraints define an instance of an IFM problem in which classical itemsets are replaced by sets of object pairs.

5 The Inverse OLAP Problem: Definition and Complexity

In this section we define a new inverse problem over a star schema that is relevant for OLAP applications.

Problem 1 (Inverse OLAP). Given a relation scheme $R(A_1, \ldots, A_n)$ with $n > 0$, the domains $D_1, \ldots, D_n$ with possibly a fixed additional number of hierarchy domains, a maximum set arity $\hat{k}$ and a set of general count constraints C on R, the *Inverse OLAP problem* consists of deciding whether there exists a relation r on R such that $C \models r$.

Proposition 2. *The Inverse OLAP problem is in* NEXP.

In the next subsection, we shall prove that the problem is indeed NEXP-complete. We point out that, in our general setting, we are considering the so-called "combined complexity" [27]: both the number of attributes, the size of constraints and the domain sizes are not fixed and are part of the input. The "program complexity" version of the problem consists of considering domain sizes as constants. On the other hand, the "data complexity" version of the inverse OLAP problem fixes the number of attributes and the size of constraints and considers domain sizes as the only problem input.

In the next sub-section we present a simple case of the Inverse OLAP problem with binary domains and prove its NEXP-hardness. It will then turn out that the "program complexity" of Inverse OLAP is NEXP-complete and, therefore, the Inverse OLAP problem is NEXP-complete under the combined complexity as well. In addition, as the constraints we shall use are well structured, we also derive that NEXP-hardness holds if we restrict our attention to well-structured constraints only.

5.1 Binary Domain Inverse OLAP

In this section we assume that all attributes $A_1, \ldots, A_n$ in U have the same binary domain $\mathcal{D} = \{0, 1\}$. We denote a k-tuple of 0-1 values by a^k and the i-th element of a^k by $a^k[i]$. We also assume that, let $X = \{A_{i_1}, \ldots, A_{i_m}\}$ be a subset of attributes in U and $\{A_{y_1}, \ldots, A_{y_k}\}$ be the attributes in $U \setminus X$, count constraints have one of the following two formats:

- *fixed tuple count constraint*, briefly denoted by (X, a^k, l, h), in the following form:
$$A_{i_1} = a^k[1] \wedge \cdots \wedge A_{i_m} = a^k[m] \rightarrow l \leq \#(\{A_{y_1}, \ldots, A_{y_k} : \mathcal{R}(A_1, \ldots, A_n)\}) \leq h$$

oarieli@mta.ac.il

– *generic tuple count constraint*, briefly denoted by (X, l, h), in the following form:

$$\forall A_{i_1} \dots A_{i_m} : \mathcal{D}(A_{i_1}) \wedge \dots \wedge \mathcal{D}(A_{i_m}) \rightarrow l \leq \#(\{A_{y_1}, \dots, A_{y_k} : \mathcal{R}(A_1, \dots, A_n)\}) \leq h$$

Example 1. Consider the binary relation R on attributes A_1, A_2, A_3, A_4, let $C_1 = (\{A_1, A_2\}, [1, 0], 1, 2)$ be a fixed tuple count constraint and $C_2 = (\{A_1, A_3\}, 1, 2)$ be generic tuple count constraint, then C_1 are C_2 are represented in the following way:

$$C_1 : A_1 = \text{"1"} \wedge A_2 = \text{"0"} \rightarrow 1 \leq \#(\{A_3, A_4 : \mathcal{R}(A_1, A_2, A_3, A_4)\}) \leq 2$$
$$C_2 : \forall A_1, A_3 : \mathcal{D}(A_1) \wedge \mathcal{D}(A_3) \rightarrow 1 \leq \#(\{A_2, A_4 : \mathcal{R}(A_1, A_2, A_3, A_4)\}) \leq 2$$

Obviously fixed tuple count constraints are well-structured; so their satisfaction can be tested in polynomial time.

Problem 2 (Binary Domain Inverse OLAP).
Given a relation scheme $R(A_1, \dots, A_n)$ on binary domains, a set of fixed tuple count constraints C on R and a set of generic tuple count constraints C' on R, the *Binary Domain Inverse OLAP* problem consists of deciding whether there exists a relation r on R such that $(C, C') \models r$.

Theorem 1. *The Binary Domain Inverse OLAP problem is* NEXP*-complete.*

Proof. Membership to NEXP immediately derives from Proposition 2.

To prove NEXP-hardness we use a reduction of the Succinct Hamiltonian Cycle Problem, that has been proved to be NEXP-complete [22,21]. In this problem a graph G_{B_C}, say with 2^n nodes, is represented in a succinct way. The nodes are coded with n bits and the edges are defined by a Boolean circuit B_C with $2n$ input and one output such that, the output value is 1 if and only if the pair of nodes described by the $2 * n$ bits of the circuit input is connected by an edge. The *Succinct Hamiltonian Cycle Problem* (SHC) is formulated as follows: given a Boolean circuit B_C with k gates, $2n$ inputs and one output, does the graph thus represented have an Hamiltonian cycle?

We next exhibit a reduction from SHC following the lines used by Kolaitis and Papadimitriou in [17] to prove that deciding whether a DATALOG program with negation has a fixpoint or not is NEXP-complete.

Given a graph G_{B_C}, say with 2^n nodes, its succinct representation is done as follows. The nodes are coded with n bits and the edges are defined by a Boolean circuit with $2n$ input and one output such that, the output value is 1 if and only if the pair of nodes described by the $2 * n$ bits of the circuit input is connected by an edge. The boolean circuit encoding the graph contains k gates and is defined as a set of quadruples $B_C = \{G_u = (u, a_u, in_u^1, in_u^2) | \ u = 1, ..., k\}$, where $k > 2n$ is the number of gates, u identifies the gate, $a_u \in \{OR, AND, NOT, IN\}$ is the kind of gate and in_u^1, in_u^2 are its input. The first $2n$ gates are of kind IN According to the value of a_u, the meaning of in_u^1 and in_u^2 change:

– if $a_u \in \{OR, AND\}$ then in_u^1 and in_u^2 represent the two inputs of the gate, encoded by the indices of the two gates, whose outputs enter a_u — for instance, $(7, AND, 5, 9)$ denotes that the inputs of the AND gate 7 are the outputs of gates 5 and 9;

oarieli@mta.ac.il

- if $a_u = NOT$ then $in_u^1 = in_u^2$ represents the unique input of the gate — for instance, $(9, NOT, 5, 5)$ denotes that the input of the NOT gate 9 is the output of gate 5;
- if $a_u = IN$ then $in_u^1 = in_u^2 = 0$ and u represents the $(i-1)$-th input bit of the circuit — for instance, $(3, IN, 0, 0)$ denotes that the input of the IN gate 3 is the second bit of the circuit (note that the input bits are numbered starting from 0 whereas the gates from 1).

Let $B_C = \{G_u = (u, a_u, in_{1,u}, in_{2,u}) | \ u = 1, ..., k\}$ be any instance of SHC with $2n$ input gates. We build in polynomial time an instance of Binary Domain Inverse OLAP as follows. The relation schema R contains the following $5n + k$ attributes:

- $A_1, \ldots, A_n, A_{n+1}, \ldots, A_{2n}$: sub-tuples on them encode pairs of nodes that are entered into the circuit inputs $0, \ldots, 2n-1$ — we shall introduce constraints such that a sub-tuple t on these attributes indicates the presence of an edge from the node coded by $t.[A_1, \ldots, A_n]$ to the node coded by $t.[A_{n+1}, \ldots, A_{2n}]$; eventually the constraints will select sub-tuples encoding an Hamiltonian circuit (if any);
- $B_1, \ldots, B_n, B_{n+1}, \ldots, B_{2n}$: sub-tuples on them encode fictitious pairs of nodes — we shall introduce constraints to construct the cycle $< (0,1), (1,2), \ldots, (2^n - 1, 2^n - 1), (2^n - 1, 0) >$, that will serve to enforce that the sub-tuples on the A_i attributes (see the previous point) form a Hamiltonian circuit; the constraints will construct the sub-tuples starting from the node 0 by implementing a simple increment operator;
- $C_1, \ldots, C_k$: sub-tuples on them encode the output values of the k gates;
- $D_1, \ldots, D_{n-1}$: sub-tuples on them encode the possible $n-1$ remainders for the increment operator (see the description for B_i attributes);
- E: the value on this attribute enables to divide a table on r into two horizontal fragments: we call them fragment 0 and fragment 1.

Let us now construct the set of fixed tuple count constraints C. They are divided into 3 groups :

Group (I): The constraints implement gate operations on the values represented in the C_i attributes, by enforcing that the wrong results cannot be present in a table - the constraints operate on the fragment 0 of a table, singled out by setting the value of column E to 0:

1. for each gate $G_u = (u, AND, in_u^1, in_u^2) \in B_C$ — the constraints enforce support 0 for the negated AND truth table (we represent the 4 constraints in parametric way):
   - $(\{C_u, C_{in_u^1}, C_{in_u^2}, E\}, \alpha, 0, 0)$, for each $\alpha = [0,1,1,0]$, $[1,0,0,0]$, $[1,1,0,0]$, $[1,0,1,0]$
2. for each gate $G_u = (u, OR, in_u^1, in_u^2) \in B_C$ — the constraints enforce support 0 for the negated OR truth table:
   - $(\{C_u, C_{in_u^1}, C_{in_u^2}, E\}, \alpha, 0, 0)$, for each $\alpha = [0,1,1,0]$, $[1,0,0,0]$, $[0,1,0,0]$, $[0,0,1,0]$
3. for each gate $G_u = (u, NOT, in_u^1, in_u^2) \in B_C$:
   - $(\{C_u, C_{in_u^1}, E\}, \alpha, 0, 0)$, for each $\alpha = [0,0,0]$, $[1,1,0]$

oarieli@mta.ac.il

4. for each gate $G_u = (u, IN, in_u^1, in_u^2) \in B_C$ — the constraints enforce the support 0 for the negated value of the $(u-1)$-th bit, that is the input to G_u:
   – $(\{C_u, A_u, E\}, \alpha, 0, 0)$, for each $\alpha = [0, 1, 0], [1, 0, 0]$
5. this constraint enforces the fragment 0 to have 2^n "yes" outputs (stored into the C_k column), so that the fragment eventually stores 2^n edges of the graph in the A_i columns :
   – $(\{C_k, E\}, [1, 0], 2^n, 2^n)$.

Group (II): This group too of constraints works on fragment 0 and enforces the construction of the fictitious cycle to be stored on the B_i attributes — given a sub-tuple on attributes $B_1, \ldots, B_n$ representing a node, say x, the constraint enforces that the sub-tuple on attributes $B_{n+1}, \ldots, B_{2n}$ of the same tuple represents the node $x+1$, by implementing an increment operator (we use the usual approach of imposing support 0 to the results that are not performed by the operator); we also observe that operator computes the node 0 as successor of the last node:

1. $(\{B_n, B_{2n}, E\}, \alpha, 0, 0)$ for each $\alpha = [1, 1, 0]$, $[0, 0, 0]$ — increment of the low order bit of the node x
2. for $i = 1, \ldots, n-1$ — increment of the other $n-1$ bits of the node x using remainders stored on attributes D_i
   – $(\{B_i, D_i, B_{n+i}, E\}, \alpha, 0, 0)$ for each $\alpha = [0, 0, 1, 0]$, $[1, 0, 0, 0]$, $[0, 1, 0, 0]$, , $[1, 1, 1, 0]$
3. $(\{B_n, D_{n-1}, E\}, \alpha, 0, 0)$ for each $\alpha = [1, 0, 0]$, $[0, 1, 0]$ — storing the remainder of the low order bit
4. for $i = 2, \ldots, n-1$ — storing the remainders of the other bits except the highest order bit, whose remainder can be neglected:
   – $(\{B_i, D_i, D_{i-1}, E\}, \alpha, 0, 0)$ for each $\alpha = [0, 0, 1, 0]$, $[1, 0, 1, 0]$, $[0, 1, 1, 0]$, , $[1, 1, 0, 0]$.

Group (III): These constraints work on fragment 1 and enforce that the two edges on A_i and B_i attributes, respectively, be self loops, i.e., respectively (x, x) and (y,y) with x not necessarily different from y — these constraints will serve for constructing a bijection between A_i nodes and B_i nodes so that, because of the bijection and of the fictitious cycle on B_i, a isomorphism is realized for which the edges on A_i form a Hamiltonian cycle (more details on the issue will be given later in the proof, here we only anticipate that the node y is actually encoded using one's complement):

1. For $i = 1, \ldots, n$:
   – $(\{A_i, A_{n+i}, E\}, \alpha, 0, 0)$ for each $\alpha = [1, 0, 1]$, $[0, 1, 1]$ — enforcing (x, x) on A_i attributes
   – $(\{B_i, B_{n+i}, E\}, \alpha, 0, 0)$ for each $\alpha = [1, 0, 1]$, $[0, 1, 1]$ — enforcing (y, y) on B_i attributes.

Let us now construct the set of generic tuple count constraints C'. There are two groups of constraints, all working on both fragments 0 and 1:

Group (IV): These constraints enforce that, for each of the two fragments 0 and 1, all nodes are stored into the A_i and B_i columns exactly once:

oarieli@mta.ac.il

1. $(\{A_1, \ldots, A_n, E\}, 1, 1)$ — first node in the pair stored on A_i attributes
2. $(\{A_{n+1}, \ldots, A_{2n}, E\}, 1, 1)$ — second node in the pair stored on A_i attributes
3. $(\{B_1, \ldots, B_n, E\}, 1, 1)$ — first node in the pair stored on B_i attributes
4. $(\{B_{n+1}, \ldots, B_{2n}, E\}, 1, 1)$ — second node in the pair stored on B_i attributes.

Group (V): These constraints, together with the ones of Group (III), realize the bijection between A_i nodes and B_i , thus enforcing the construction of a Hamiltonian circuit — they work for the whole table and not separately for each of the two fragments as in the Group (IV):

1. For each B_j with $j = 1, \ldots, n$
   $(\{A_1, \ldots, A_n, B_j\}, 1, 1)$ — a first node in a pair stored on A_i attributes (say x) is associated to exactly one of the first nodes in the pairs stored on B_i attributes, say y; moreover, since x is present once in fragment 0 and once in fragment 1, the association inside fragment 1 is between x and the one's complement of y in order to respect the constraints;
2. For each B_i with $i = n + 1, \ldots, 2n$
   $(\{A_{n+1}, \ldots, A_{2n}, B_i\}, 1, 1)$ — association for the second node in a pair stored on A_i attributes; note that, because of constraints of Group (III), the association for the node is the same as the one enforced when the node appears as first node in a pair (see constraints 1 of this group).

As mentioned before, the constraints enforces that a table r on R has 2^{n+1} tuples and can be segmented by a selection on E into fragments 0 and 1, each with 2^n tuples: $F_0 = \sigma_{E=0}(r)$ and $F_1 = \sigma_{E=1}(r)$. Let us now further consider the following sub-fragments of r obtained by suitably projecting the fragments F_0 and F_1:

$\mathbf{A} = \pi_{\mathbf{A_1},\ldots,\mathbf{A_{2n}}}(\mathbf{F_0})$ $\mathbf{B} = \pi_{\mathbf{A_1},\ldots,\mathbf{A_{2n}}}(\mathbf{F_0})$
$\mathbf{A_1} = \pi_{\mathbf{A_1},\ldots,\mathbf{A_n}}(\mathbf{A})$ $\mathbf{A_2} = \pi_{\mathbf{A_{n+1}},\ldots,\mathbf{A_{2n}}}(\mathbf{A})$
$\mathbf{B_1} = \pi_{\mathbf{B_1},\ldots,\mathbf{B_n}}(\mathbf{B})$ $\mathbf{B_2} = \pi_{\mathbf{B_{n+1}},\ldots,\mathbf{B_{2n}}}(\mathbf{B})$
$\mathbf{A'} = \pi_{\mathbf{A_1},\ldots,\mathbf{A_{2n}}}(\mathbf{F_1})$ $\mathbf{B'} = \pi_{\mathbf{A_1},\ldots,\mathbf{A_{2n}}}(\mathbf{F_1})$
$\mathbf{A'_1} = \pi_{\mathbf{A_1},\ldots,\mathbf{A_n}}(\mathbf{A'})$ $\mathbf{A'_2} = \pi_{\mathbf{A_{n+1}},\ldots,\mathbf{A_{2n}}}(\mathbf{A'})$
$\mathbf{B'_1} = \pi_{\mathbf{B_1},\ldots,\mathbf{B_n}}(\mathbf{B'})$ $\mathbf{B'_2} = \pi_{\mathbf{B_{n+1}},\ldots,\mathbf{B_{2n}}}(\mathbf{B'})$

The sub-fragments $\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{B_1}$, $\mathbf{B_2}$, $\mathbf{A'_1}$, $\mathbf{A'_2}$, $\mathbf{B'_1}$ and $\mathbf{B'_2}$ are depicted in Figure 1:

A_1.. ..A_n	A_{n+1}.. ..A_{2n}	B_1.. ..B_n	B_{n+1}.. ..B_{2n}	L.. ..C_k D_1.. ..D_{n-1}	E
$\mathbf{A_1}$	$\mathbf{A_2}$	$\mathbf{B_1}$	$\mathbf{B_2}$		0 .. 0 .. 0
$\mathbf{A'_1}$	$\mathbf{A'_2}$	$\mathbf{B'_1}$	$\mathbf{B'_2}$		1 .. 1 .. 1

Fig. 1. Table r on R

oarieli@mta.ac.il

Each of sub-fragments $\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{B_1}$, $\mathbf{B_2}$, $\mathbf{A'_1}$, $\mathbf{A'_2}$, $\mathbf{B'_1}$ and $\mathbf{B'_2}$ has n columns and 2^n tuples and stores all the 2^n nodes of the graph.

We have the following claims — with a little abuse of notation, given a tuple t and a fragment X, t_X will denote the projection of t on the attributes of X:

Claim. 2 *For each tuple t in $\mathbf{A}$, the sub-tuples $t_{\mathbf{A_1}}$ and $t_{\mathbf{A_2}}$ represent two vertices u, v of the succinct graph such that there exists an edge between u and v in the graph.*

Proof. The *count constraints* of group (I) implements the Boolean Circuit and the constraint 4 of the group imposes that the circuit must have output value 1, thus there is an edge between u and v. □

Claim. 3 *For each tuple t in $\mathbf{B}$, $\mathrm{int}(t_{\mathbf{B_2}}) = \mathrm{int}(t_{\mathbf{B_1}} + 1) \bmod 2^n$ where $\mathrm{int}(t_{\mathbf{B_1}})$ and $\mathrm{int}(t_{\mathbf{B_2}})$ are the non negative integer values represented by the binary tuples $t_{\mathbf{B_1}}$ and $t_{\mathbf{B_2}}$, respectively.*

Proof. The count constraints of group (II) implement a binary incremental circuit that take in input a tuple $t_{\mathbf{B_1}}$ and returns the tuple $t_{\mathbf{B_2}}$ such that $\mathrm{int}(t_{\mathbf{B_2}}) = (\mathrm{int}(t_{\mathbf{B_1}}) + 1) \bmod 2^n$. □

Claim. 4 *Let $t \in F_0$ and $t' \in F_1$. Then:*
1. *if $t_{\mathbf{A_1}} = t'_{\mathbf{A'_1}}$ then $t'_{\mathbf{B'_1}} = \mathrm{neg}(t_{\mathbf{B_1}})$ where $\mathrm{neg}(t_{\mathbf{B_1}})$ is the one's complement of tuple $t_{\mathbf{B_1}}$,*
2. *if $t_{\mathbf{A_2}} = t'_{\mathbf{A'_2}}$ then $t'_{\mathbf{B'_2}} = \mathrm{neg}(t_{\mathbf{B_2}})$.*

Proof. The proof is a direct consequence of the generic tuple count constraints of group (V). □

Claim. 5 *For each tuple $t \in F_1$, $t_{\mathbf{A'_1}} = t_{\mathbf{A'_2}}$ and $t_{\mathbf{B'_1}} = t_{\mathbf{B'_2}}$.*

Proof. The proof is a direct consequence of the count constraints of group (III). □

By Claim 2, the tuples in $\mathbf{A}$ represent 2^n distinct edges of the graph G_{B_C} such that each node has exactly two incident edges. However, this condition is not sufficient to establish the existence of Hamiltonian Cycle because it could happen that G consists of several Hamiltonian sub-cycles.

To actually enforce that the edges in $\mathbf{A}$ form a Hamiltonian cycle, we have constructed a fictitious Hamiltonian cycle in $\mathbf{B}$ and defined a bijection between nodes in $\mathbf{A}$ and in $\mathbf{B}$ so to have a isomorphism between the two cycles. By Claim 3, $\mathbf{B}$ represents a lexicographic cycle from the node 0 to $2^n - 1$ and back to 0 — we stress that this cycle is not in general present in the graph.

By Claims 4 and 5, there must be a isomorphism between the edges in $\mathbf{A}$ and $\mathbf{B}$ and this implies that the graph must have a Hamiltonian circuit. We can then conclude that the succinct graph G_{B_C} has a Hamiltonian circuit if and only if there exists a relation r on R that satisfies the count constraints in C and C'. This concludes the proof. □

From this theorem we immediately derive the following important corollary.

Corollary 1. *The* Inverse OLAP *problem is* NEXP *complete under both the scheme complexity and the combined complexity.* □

oarieli@mta.ac.il

5.2 Binary Attribute Inverse OLAP

Let us now assume that U contains only two attributes, say $U = \{A_T, A_I\}$ with domains T and $\mathcal{I} = \{o_1, \ldots, o_n\}$. Let Y denote any given subset of $\mathcal{I}$. We assume that a set of itemsets $S = \{S_1, \ldots, S_m\}$ is given. Then count constraints have one of the following four formats:

- *k-support count constraint*, briefly denoted by $\sigma(Y, k, l, h)$, where Y is an itemset, k, l and h are non negative integers with $0 < k \leq |Y|$ and $l \leq h$ that expresses

$$\forall I' : I' \subseteq Y \wedge S_1 \not\subset I' \wedge I' \not\subset S_1 \wedge \ldots \wedge S_m \not\subset I' \wedge I' \not\subset S_m \wedge \#(I') = k \rightarrow \\ l \leq \#(\{T : I' \subseteq \{I : \mathcal{R}(T, I)\}\}) \leq h;$$

- *k-duplicate count constraint*, briefly denoted by $\delta(Y, k, l, h)$, where Y, k, l and h are defined as above:

$$\forall I' : I' \subseteq Y \wedge \wedge \#(I') = k \rightarrow l \leq \#(\{T : I' = \{I : \mathcal{R}(T, I)\}\}) \leq h.$$

Problem 3 (Binary Attribute Inverse OLAP).
Given a relation scheme $R = \{A_T, A_I\}$ on domains T and $\mathcal{I}$, a set of itemsets S, a set of k-support count constraints Γ_σ on R and a set of k-duplicate count constraints Γ_δ on R, the *Binary Attribute Inverse OLAP* problem consists of deciding whether there exists a relation r on R such that $(\Gamma_\sigma, \Gamma_\delta) \models r$. □

Theorem 2. *The Binary Attribute Inverse OLAP is* NEXP*-complete.*

In the proof, reported in [25], we exhibit a reduction from Binary Domain Inverse OLAP.

5.3 Data Complexity of Inverse OLAP

In this section we will define a special case of Inverse OLAP problem, called Fixed Hierarchy Inverse OLAP, whose input instance is formed by a fixed number of hierarchy domains. The sizes of the schema relation and of the count constraints are fixed. This problem is useful in order to establish the Data Complexity of Inverse OLAP.

Problem 4 (Fixed Hierarchy Inverse OLAP).
Let R be a relation scheme of form

$$R(C_\sigma, Y_\sigma, K_\sigma, L_\sigma, H_\sigma, C_\delta, Y_\delta, K_\delta, L_\delta, H_\delta, S^1, S^2, T, I),$$

$\mathcal{D}_{C_\sigma,Y_\sigma}$, $\mathcal{D}_{C_\sigma,K_\sigma}$, $\mathcal{D}_{C_\sigma,H_\sigma}$, $\mathcal{D}_{C_\sigma,L_\sigma}$, $\mathcal{D}_{C_\delta,Y_\delta}$, $\mathcal{D}_{C_\delta,K_\delta}$, $\mathcal{D}_{C_\delta,H_\delta}$, $\mathcal{D}_{C_\delta,L_\delta}$, $\mathcal{D}_{S^1,S^2}$, be the hierarchy domains, and C be a set of two count constraints of constant size, the *Binary Attribute Inverse OLAP* problem consists of deciding whether there exists a relation r on R such that $C \models r$. □

Theorem 3. *The Fixed Hierarchy Inverse OLAP problem is* NEXP*-complete.*

In the proof, reported in [25], we exhibit a reduction from Binary Attribute Inverse OLAP. Since the schema relation and the count constraints are fixed in Fixed Hierarchy Inverse OLAP, then the following corollary holds.

Corollary 2. *The Inverse OLAP problem is* NEXP*- complete under data complexity.*

oarieli@mta.ac.il

6 A Step towards Aggregate Data Exchange

In this sections we present some meaningful example of how count constraints can be exploited for aggregate data exchange. We first recall the classical data exchange setting: $(S, T, \Sigma_{st}, \Sigma_t)$, where S is the source relational scheme, T is the target relational database scheme, Σ_t is a logical formula on T and Σ_{st} are source-to-target dependencies of the form $\forall \mathbf{X}(\phi_S(\mathbf{X}) \rightarrow \chi_T(\mathbf{X}))$, where $\phi_S(\mathbf{X})$ and $\chi_T(\mathbf{X})$ are formula on S and T, respectively.

Aggregate data exchange for preserving privacy
The target relational database scheme consists of a unique relation scheme, that is the one used in Section 4: $\mathcal{R}(T, I, B, S, A)$. Recall that the meaning of the attributes is: T (Transaction), I (Item), B (Brand), S (Store), A (Area), and that the following FDs hold: $T \rightarrow S$, $S \rightarrow A$.

The source relational database scheme consists of three relation schemes: $\mathcal{TR}(T, I, B)$, $\mathcal{ST}(T, S)$ and $\mathcal{AR}(S, A)$. Observe that this scheme is the normalized version of the target scheme.

We want the target relation to be the natural join of the source relation but, for privacy reasons, the associations between transactions and pairs of item and brand must be perturbed: the transactions IDs of the same store are permuted. For instance, if the store s has n transactions $t_1, \ldots, t_n$, the block of item-brand pairs of a transaction t_i are moved to a transaction t_j, then the block of t_j is moved to another transaction and so on.

Let us first use the classical setting (that includes the usage of an existential quantifier) to implement two natural joins of the source relations instead of only one so that we can later perform a permutation of transactions inside every store:

$$\forall x_T, x_I, x_B, x_S, x_A(\mathcal{TR}(x_T, x_I, x_B) \wedge \mathcal{ST}(x_T, x_S) \wedge \mathcal{AR}(x_S, x_A) \rightarrow \exists \hat{x}_T \mathcal{R}(\hat{x}_T, I, B, S, A));$$
$$\forall x_T, x_S, \ (\ \mathcal{ST}(x_T, x_S) \ \rightarrow \exists x_I, x_B, x_A \ \mathcal{R}(x_T, x_I, x_B, x_S, x_A) \ ).$$

We now use a count constraint to enforce that the total number of tuples in $\mathcal{TR}$ is equal to the total number of tuples in $\mathcal{R}$ so that the target relation cannot store additional tuples:

$$\forall s \ (\ s = \#(\{x_T, x_I, x_B : \mathcal{TR}(x_T, x_I, x_B)\}) \ \rightarrow$$
$$s = \#(\{x_T, x_I, x_B, x_S, x_A : \ \mathcal{R}(x_T, x_I, x_B, x_S, x_A)\}) \ ).$$

So we have lost the correspondence between transactions in the source and in the target scheme; but the following constraint imposes that the original structure of transactions is preserved modulo permutation of transactions IDs:

$$\forall x_S, x_T, s \ (\ \mathcal{ST}(x_T, x_S) \wedge s = \{x_I, x_B : \mathcal{TR}(x_T, x_I, x_B)\}) \ \rightarrow$$
$$\exists \hat{x}_T \ (\ s = \{x_I, x_B : \forall A \ \mathcal{R}(\hat{x}_T, x_I, x_B, x_S, x_A)\} \ ) \ ).$$

The above constraint can be equivalently rewritten using a count predicate instead of the existential quantifier:

$$\forall x_S, x_T, s \ (\ \mathcal{ST}(x_T, x_S) \wedge s = \{x_I, x_B : \mathcal{TR}(x_T, x_I, x_B)\}) \ \rightarrow$$
$$1 \leq \#(\{\hat{x}_T : \ s = \{x_I, x_B : \forall A \ \mathcal{R}(\hat{x}_T, x_I, x_B, x_S, x_A)\} \ \} \ ) \ ).$$

Thus the count predicates is able to emulate an existential quantifier.

oarieli@mta.ac.il

Data exchange to an OLAP scheme
We now assume that the relation $\mathcal{R}(T, I, B, S, A)$ represents the source scheme. The target scheme is an OLAP scheme $\mathcal{SN}(S, I, B, N)$ that, for every store, represents in N the total number of item-brand pairs that are in all transactions of that store.

We aggregate data in the target relation using a count predicate in the following constraint:

$$\forall x_I, x_B, x_S, x_A, x_N \ (\ x_N = \#(\{x_T : \ \mathcal{R}(x_T, x_I, x_B, x_S, x_A) \ \rightarrow \mathcal{SN}(x_S, x_I, x_B, x_N) \ ).$$

A final count constraint imposes that the target relation cannot store additional tuples:

$$\forall s \ (\ s = \#(\{x_S, x_I, x_B : \ \forall x_T, x_A \ \mathcal{R}(x_T, x_I, x_B, x_S, x_A) \ \rightarrow \\ s = \#(\{x_S, x_I, x_B : \ \forall x_N \ \mathcal{SN}(S, I, B, x_N)\} \ ) \ ).$$

7 Conclusion

We have introduced a new type a constraints, called count constraints, a new inverse mining problem, called Inverse OLAP, that is a powerful extension of Inverse Frequent itemsets Mining: given a star schema and a number of count constraints, does there exist a satisfying relation? The new problem turns out to be NEXP complete under various conditions: combined complexity, program complexity and data complexity. We have also shown that our setting for expressing count constraints can be used for performing aggregate data exchange.

We conclude by mentioning that, despite the high complexity of the Inverse OLAP problem, an approximate solution can be found in a limited amount of time in some practical situations even for large instances, by adopting and extending classical techniques used for solving large-scale linear programming. In [16] one of such techniques, called column-generation linear programming, is applied to solve an IFM problem (indeed non just the decision problem but the actual construction of a satisfying transaction database) very similar to the binary attribute OLAP and is capable of handling instances with several hundreds of items.

References

1. Afrati, F.N., Kolaitis, P.G.: Answering aggregate queries in data exchange. In: PODS, pp. 129–138 (2008)
2. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: SIGMOD Conference, pp. 207–216 (1993)
3. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: ACM SIGMOD, pp. 439–450 (2000)
4. Arasu, A., Kaushik, R., Li, J.: Data generation using declarative constraints. In: Proceedings of the 2011 International Conference on Management of Data, SIGMOD 2011, pp. 685–696. ACM, New York (2011)
5. Arenas, M., Barcel, P., Fagin, R., Libkin, L.: Locally consistent transformations and query answering in data exchange. In: PODS 2004, pp. 229–240 (2004)
6. Beeri, C., Vardi, M.Y.: Polynomial-time implication problems for unary inclusion dependencies. J. of the ACM 37, 15–46 (1990)

oarieli@mta.ac.il

7. Binnig, C., Kossmann, D., Lo, E., Özsu, M.T.: Qagen: generating query-aware test databases. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of data, SIGMOD 2007, pp. 341–352. ACM, New York (2007)
8. Calders, T.: Computational complexity of itemset frequency satisfiability. In: PODS, pp. 143–154 (2004)
9. Calders, T.: The complexity of satisfying constraints on databases of transactions. Acta Inf. 44(7-8), 591–624 (2007)
10. Chaudhuri, S., Dayal, U.: An overview of data warehousing and olap technology. SIGMOD Record 26(1), 65–74 (1997)
11. Cosmadakis, S.S., Kanellakis, P.C., Vardi, M.Y.: Polynomial-time implication problems for unary inclusion dependencies. J. of the ACM 37, 15–46 (1990)
12. Faber, W., Pfeifer, G., Leone, N., Dell'Armi, T., Ielpa, G.: Design and implementation of aggregate functions in the dlv system. TPLP 8(5-6), 545–580 (2008)
13. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data Exchange: Semantics and Query Answering. In: Calvanese, D., Lenzerini, M., Motwani, R. (eds.) ICDT 2003. LNCS, vol. 2572, pp. 207–224. Springer, Heidelberg (2002)
14. Fagin, R., Kolaitis, P.G., Popa, L.: Data exchange: getting to the core. ACM Trans. Database Syst. 30(1), 174–210 (2005)
15. Golfarelli, M., Rizzi, S.: Data Warehouse Design: Modern Principles and Methodologies. Mac Graw Hill (2009)
16. Guzzo, A., Moccia, L., Saccà, D., Serra, E.: Solving inverse frequent itemset mining with infrequency constraint via large-scale linear programs (2011), http://sacca.deis.unical.it/#view=object&format=object&id=981/gid=160
17. Kolaitis, P.G., Papadimitriou, C.H.: Why not negation by fixpoint? In: PODS, pp. 231–239 (1988)
18. Lenzerini, M.: Data integration: A theoretical perspective. In: PODS, pp. 233–246 (2002)
19. Lo, E., Cheng, N., Hon, W.-K.: Generating databases for query workloads. Proc. VLDB Endow. 3, 848–859 (2010)
20. Mielikainen, T.: On inverse frequent set mining. In: Proc. of 2nd Workshop on Privacy Preserving Data Mining (PPDM), pp. 18–23. I. C. Society (2003)
21. Papadimitriou, C.H.: Computational complexity. Addison-Wesley, Reading (1994)
22. Papadimitriou, C.H., Yannakakis, M.: A note on succinct representations of graphs. Information and Control 71(3), 181–185 (1986)
23. Ramesh, G., Maniatty, W., Zaki, M.J.: Feasible itemset distributions in data mining: theory and application. In: PODS, pp. 284–295 (2003)
24. Rosati, R.: On the decidability and finite controllability of query processing in databases with incomplete information. In: PODS, pp. 356–365 (2006)
25. Saccà, D., Guzzo, A., Serra, E.: Appendix to this paper (2012), http://sacca.deis.unical.it/#view=object&format=object&id=960/gid=160
26. Syrjänen, T.: Logic Programs and Cardinality Constraints: Theory and Practice. Doctoral dissertation, TKK Dissertations in Information and Computer Science TKK-ICS-D12, Helsinki University of Technology, Department of Information and Computer Science (2009)
27. Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: STOC, pp. 137–146 (1982)
28. Wu, X., Wu, Y., Wang, Y., Li, Y.: Privacy aware market basket data set generation: A feasible approach for inverse frequent set mining. In: Proc. 5th SIAM International Conference on Data Mining (2005)
29. Zhang, X., Ozsoyoglu, Z.M.: Implication and referential constraints: A new formal reasoning. IEEE Trans. on Knowledge and Data Engineering 9, 894–910 (1997)

oarieli@mta.ac.il

Synchronous Parallel Database Transformations

Klaus-Dieter Schewe[1] and Qing Wang[2]

[1] Software Competence Center Hagenberg and Johannes-Kepler-University Linz, Austria
kd.schewe@scch.at, kd.schewe@faw.at, kdschewe@acm.org
[2] Department of Information Science, University of Otago, New Zealand
qing.wang@otago.ac.nz

Abstract. The DB-ASM thesis states that every database transformation can be expressed by a variant of Abstract State Machines. These machines permit unbounded parallelism only on the finite database part of a state. This paper generalises this work by permitting unbounded parallelism on the algorithmic part of the state as well. The "parallel DB-ASM"-thesis results from combining Gurevich's parallel ASM thesis with the DB-ASM thesis. In doing so, it turns out that the postulates for synchronous parallel database transformations can be significantly simplified compared with the seminal work of Gurevich. The key idea is to generalise the notion of bounded exploration witnesses allowing them to include special non-ground terms.

Keywords: Abstract State Machine, database transformation, bounded exploration, synchronous parallelism.

1 Introduction

Abstract State Machines (ASMs) has established themselves as an elegant and powerful instrument in the development of software systems [3]. However, their origin goes back to the fundamental question to give language-independent definition of the notion of (sequential) algorithm. So, the "ASM thesis" is considered to be an analogue to Church's thesis, which clarified the notion of computable function.

The ASM thesis actually refers to more than one result. The latest form of the sequential ASM thesis refers to Gurevich's seminal work on sequential algorithms, which he defined by three simple, intuitive postulates [6], whereas the parallel ASM thesis refers to the generalisation by Blass and Gurevich to (synchronous) parallel algorithms [2].

In our own previous work we adapted the sequential ASM thesis to characterise database transformations in general by a variant of ASMs called DB-ASMs [7]. The core of the approach is similar to Gurevich's seminal work: provide a language-independent definition through a set of intuitive postulates, define formally an abstract machine model, and prove that the postulates are exactly captured by the machine model. On these grounds it was then possible to tailor

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 370–383, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

the model to specific data model, e.g. XML [8], and to pick up partial updates [9] thereby extending previous work on applicative algebras in [5].

This paper is dedicated to the problem to remove the restriction that only sequential algorithms are exploited in the DB-ASM thesis. That is, we want to generalise our previous work to the case of synchronous parallel database transformations in general. The corresponding parallel ASM thesis [2] is significantly more complicated than the sequential sequential ASM thesis, as it involves several postulates to handle the synchronisation of different parallel branches of a computation.

We first review the DB-ASM thesis in Section 2. Then, Section 3 contains the core of the paper addressing the formal generalisation to parallel synchronous database ASMs. We conclude with a brief summary and outlook.

2 The Database-ASM Thesis

In our work on DB-ASMs [7] we adopted the approach of Gurevich that led to the sequential ASM thesis [6], i.e. we defined database transformations in a language-independent way by five intuitive postulates, which we shall briefly review in this section. We also present DB-ASMs, the variant of ASMs that captures exactly these database transformations.

2.1 Database Transformations

We will formally introduce the five postulates for database transformations: the *sequential time postulate*, the *abstract state postulate*, the *background postulate*, the *bounded exploration postulate*, and the *bounded non-determinism postulate*. An object satisfying these postulates will be called a *database transformation*.

As in [6] we assume that a database transformation same as any algorithm proceeds step-wise on a set of states. This leads to the following *sequential state postulate*, which (with some minor modifications) is the same as in the Gurevich's ASM thesis.

Postulate 1 (sequential time postulate). A database transformation t is associated with a non-empty set of states $\mathcal{S}_t$ together with non-empty subsets $\mathcal{I}_t$ and $\mathcal{F}_t$ of initial and final states, respectively, and a one-step transition relation τ_t over $\mathcal{S}_t$, i.e. $\tau_t \subseteq \mathcal{S}_t \times \mathcal{S}_t$.

The second postulate concerns the states. Its first part is a simple adaptation of the abstract state postulate in [6] to the presence of final states and the fact that one-step transition is a binary relation. The second part is specific to databases. The rationale is that the states of a database transformation should capture databases that are defined over some database schema.

Postulate 2 (abstract state postulate). All states $S \in \mathcal{S}_t$ of a database transformation t are structures over the same signature Σ_t, and whenever $(S, S') \in \tau_t$ holds, the states S and S' have the same base set B. The sets $\mathcal{S}_t$, $\mathcal{I}_t$ and $\mathcal{F}_t$

oarieli@mta.ac.il

are closed under isomorphisms, and for $(S_1, S_1') \in \tau_t$ each isomorphism σ from S_1 to S_2 is also an isomorphism from S_1' to $S_2' = \sigma(S_1')$ with $(S_2, S_2') \in \tau_t$.

Furthermore, the signature Σ_t is composed as a disjoint union out of a signature Σ_{db} (called *database part*), an signature Σ_a (called *algorithmic part*), and a finite set of bridge function symbols each with a fixed arity, i.e. $\Sigma_t = \Sigma_{db} \cup \Sigma_a \cup \{f_1, \ldots, f_\ell\}$. The base set of a state S is $B = B_{db}^{ext} \cup B_a$ with interpretation of function symbols in Σ_{db} and Σ_a over $B_{db} \subseteq B_{db}^{ext}$ and B_a, respectively, with B_{db} depending on the state S. The interpretation of a bridge function symbol of arity k defines a function from B_{db}^k to B_a. With respect to such states S the restriction to Σ_{db} is a finite structure, i.e. B_{db} is finite.

As states are structures, terms t with variables $fr(t) = \{x_1, \ldots, x_n\}$ can be evaluated in them for a given variable assignment $\zeta = \{x_1 \mapsto a_1, \ldots, x_n \mapsto a_n\}$ with elements $a_1, \ldots, a_n$ of the base set of the structure. We use the notation $val_{S,\zeta}(t)$ for the value in the base set of the state S that results from evaluating term t in S with respect to ζ.

The third postulate refers to the background of a computation, which contains everything that is needed to perform the computation, but is not yet captured by the state. For instance, truth values and their connectives, and a value $\perp$ to denote undefinedness constitute necessary elements in a background. The background is the same as in [2].

Postulate 3 (background postulate). Each state of a database transformation t must contain

- an infinite set of reserve values,
- truth values and their connectives, the equality predicate, the undefinedness value $\perp$, and
- a background class $\mathcal{K}$ defined by a background signature V_K that contains at least a binary tuple constructor $(\cdot)$, a multiset constructor $\langle\cdot\rangle$, and function symbols for the following operations: pairing and projection for pairs, empty multiset $\langle\rangle$, singleton $\langle x\rangle$, binary multiset union $\uplus$, general multset union $\biguplus x$, $AsSet$, and $\mathbf{I}x$ ("the unique") on multisets.

The bounded exploration postulate in the sequential ASM thesis in [6] uses a finite set of ground terms as bounded exploration witness in the sense that whenever states S_1 and S_2 coincide over this set of ground terms the update set produced by the sequential algorithm is the same in these states. The intuition behind the postulate is that only the part of a state that is given by means of the witness will actually be explored by the algorithm.

The fact that only finitely many locations can be explored remains the same for database transformations. However, permitting parallel accessibility within the database part of a state forces us to slightly change our view on the bounded exploration witness. For this we need access terms.

Definition 1. An *access term* is either a ground term α or a triple (f, β, α) of terms, the variables $x_1, \ldots, x_n$ in which refer to the arguments of $f \in \Sigma_{db}$. The interpretation of (f, β, α) in a state S is the set of locations

oarieli@mta.ac.il

$$\{f(a_1, \ldots, a_n) \mid val_{S,\zeta}(\beta) = val_{S,\zeta}(\alpha) \text{ with } \zeta = \{x_1 \mapsto a_1, \ldots, x_n \mapsto a_n\}\}.$$

Structures S_1 and S_2 *coincide* over a set T of access terms iff the interpretation of each $\alpha \in T$ and each $(f, \beta, \alpha) \in T$ over S_1 and S_2 are equal.

For a database transformation t, a function symbol f of arity n in the state signature and values $a_1, ..., a_n, v$ in the base set of states $f(a_1, ..., a_n)$ is called a *location*. An *update* is a pair (ℓ, v), where ℓ is a location. $\Delta(t, S)$ denotes the set of updates that t will perform in state S (this is well defined [7]).

Due to our request that the database part of a state is always finite there will be a maximum number m of elements that are accessible in parallel. Furthermore, there is always a number n such that n variables are sufficient to describe the updates of a database transformation, and n can be taken to be minimal. Then for each state S the upper boundary of exploration is $\mathcal{O}(m^n)$, where m depends on S. Taking these together we obtain our fourth postulate.

Postulate 4 (bounded exploration postulate). For a database transformation t there exists a fixed, finite set T of access terms of t (called *bounded exploration witness*) such that $\Delta(t, S_1) = \Delta(t, S_2)$ holds whenever the states S_1 and S_2 coincide over T.

The last postulate bounds non-determinism in a database transformation to choices depending on the database part.

Postulate 5 (bounded non-determinism postulate). For a database transformation t, if there are states S_1, S_2 and $S_3 \in \mathcal{S}_t$ with $(S_1, S_2) \in \tau_t$, $(S_1, S_3) \in \tau_t$ and $S_2 \neq S_3$, then there exists a non-ground access term of the form (β, α) in the bounded exploration witness of t.

In particular, if we ignore the database part by assuming $\Sigma_d b = \emptyset$, then a database transformations degenerates to a sequential algorithm in the sense of Gurevich [6]. There would be no non-determinism according to Postulate 5, and bounded exploration witnesses would only contain ground terms.

2.2 Database Abstract State Machines

As for ASMs, DB-ASMs are defined by rules which we sketch here (for formal details see [7]):

- If $t_0, \ldots, t_n$ are terms over Σ, and f is a n-ary function symbol in Σ, then $f(t_1, \ldots, t_n) := t_0$ is a rule r in $\mathcal{R}$ called *assignment rule*.
- If φ is a Boolean term and $r' \in \mathcal{R}$ is a DB-ASM rule, then **if** φ **then** r' **endif** is a rule r in $\mathcal{R}$ called *conditional rule*.
- If φ is a Boolean term with only database variables, $\{x_1, \ldots, x_k\} \subseteq fr(\varphi)$ and $r' \in \mathcal{R}$ is a DB-ASM rule, then **forall** $x_1, \ldots, x_k$ **with** φ **do** r' **enddo** is a rule r in $\mathcal{R}$ called *forall rule*.
- If $r_1, \ldots, r_n$ are rules in $\mathcal{R}$, then the rule r defined as **par** $r_1 \ldots r_n$ **endpar** is a rule in $\mathcal{R}$, called *parallel rule*.

oarieli@mta.ac.il

- If φ is a Boolean term with only database variables, $\{x_1, \ldots, x_k\} \subseteq fr(\varphi)$ and $r' \in \mathcal{R}$ is a DB-ASM rule, then **choose** $x_1, \ldots, x_k$ **with** φ **do** r' **enddo** is a rule r in $\mathcal{R}$ called *choice rule*.
- If r_1, r_2 are rules in $\mathcal{R}$, then the rule r defined as **seq** $r_1 \;\; r_2$ **endseq** is a rule in $\mathcal{R}$, called *sequence rule*.
- If $r' \in \mathcal{R}$ is a DB-ASM rule and θ is a location function that assigns location operators ρ to terms t with $var(t) \subseteq fr(r')$, then **let** $\theta(t) = \rho$ **in** r' **endlet** is a rule $r \in \mathcal{R}$ called *let rule*.

Here a location function ρ on the set $M(D)$ of all non-empty multisets over a domain D is defined by a unary function $f_\alpha : D \to D$, a commutative and associative binary operation $\odot$ over D, and a unary function $f_\beta : D \to D$, which define $\rho(m) = f_\beta(f_\alpha(b_1) \odot \cdots \odot f_\alpha(b_n))$ for $m = \langle b_1, ..., b_n \rangle \in M(D)$.

The key results on database transformations are the following two theorems (for a proof see [7]):

Theorem 1. *Each DB-ASM $\mathcal{M}$ defines a database transformation t with the same signature and background as $\mathcal{M}$.*

Theorem 2. *For every database transformation t there exists an equivalent DB-ASM $\mathcal{M}$ with the same background.*

3 Deterministic Synchronous Parallel Database Transformations

In a nutshell the model of DB-ASMs combines computations on finite databases with any sequential algorithm working on the algorithmic part of the states. This generalises reflective relational machines (see [1]) in that computations refer to sequential ASMs rather than Turing machines. As shown in [8] for the case of XML the access to the database part can be enhanced by adding more complex logical constructs.

However, the algorithms incorporated into DB-ASMs are only sequential in the sense that there is no way to have arbitrarily many parallel branches in a computation, the number of which is not determined a priori by the algorithm, but depends on the state. Our aim is to remove this restriction.

3.1 Modified Postulates

One way to do this would be to redo the work that led to the postulates for database transformations starting from the postulates for parallel algorithms in [2] instead of those for sequential algorithms in [6]. However, we will proceed differently by first shedding more light into the extensions in our postulates compared with the sequential ASM thesis:

- The key difference in the sequential time postulate is the support for non-determinism. If for the time being we concentrate on deterministic database transformations, we could have kept Gurevich's postulate without change.

oarieli@mta.ac.il

- As a consequence, there would be no need for the bounded non-determinism postulate.
- The use of meta-finite states in the abstract state postulates is not more than a restriction needed for databases, and the background postulates only makes the need for certain values and operations obvious.
- So, the key difference lies in the extension of the bounded exploration witnesses.

Checking the proofs for the DB-ASM thesis, i.e. for Theorems 1 and 2, shows that the restriction of access terms to the database part of the structure is not really necessary. Therefore, we will define *(deterministic, synchronous) parallel database transformations* (parDB transformations for short) by four postulates: a slightly modified sequential time postulate 6, the abstract state postulate 2, the background postulate 3, and a new bounded exploration postulate 4, in which the notion of access term is generalised.

Postulate 6 (sequential time postulate). A database transformation t is associated with a non-empty set of states $\mathcal{S}_t$ together with a non-empty subset $\mathcal{I}_t$ of initial states, and a one-step transition function $\tau_t : \mathcal{S}_t \to \mathcal{S}_t$.

For the bounded exploration postulate we only have to generalise the notion of access term letting comprehensions not only refer to the database part of the signature.

Definition 2. An *access term* is either a ground term α or a triple (f, β, α) of terms, the variables $x_1, \ldots, x_n$ in which refer to the arguments of $f \in \Sigma$. The interpretation of (f, β, α) in a state S is the set of locations

$$\{f(a_1, \ldots, a_n) \mid val_{S,\zeta}(\beta) = val_{S,\zeta}(\alpha) \text{ with } \zeta = \{x_1 \mapsto a_1, \ldots, x_n \mapsto a_n\}\}.$$

Structures S_1 and S_2 *coincide* over a set T of access terms iff the interpretation of each $\alpha \in T$ and each $(\beta, \alpha) \in T$ over S_1 and S_2 are equal.

3.2 Examples

Let us now look at some examples for synchronous parallel database transformations. In the following we look at cases where there is obviously unbounded parallelism involved in the sense that the parallel branches are not determined by the algorithm, but by the state (for a detailed discussion see [2]). Thus, we emphasize that these algorithms are captured by the generalised bounded exploration postulate involving non-ground terms in the bounded exploration witnesses.

Example 1. Let us look at the example of graph inversion, which was also used in [2] to motivate the postulates for parallel algorithms. Here we can assume two database relations in the signature Σ_{db}, the unary relation NODE, and the binary relation EDGE (let us simplify the example by ignoring the bridge functions). The algorithm can be simply expressed by the rule

oarieli@mta.ac.il

forall x, y **with** $\textsc{Node}(x) \wedge \textsc{Node}(y)$
do
$\quad \textsc{Edge}(x, y) := \neg\textsc{Edge}(x, y)$
enddo

Here the exploration boundary witness can be taken as

$$T = \{(\textsc{Edge}, \textsc{Node}(x), \textsc{Node}(y))\}.$$

Example 2. Take the rather well known LCR algorithm for leader election in a ring. Again, let us keep the presentation simple by ignoring the bridge functions. Then we need a unary function UID, which maps node identifiers onto numerical values, a unary function SEND, which does the same, an incoming message function IN that maps node identifiers to numerical values or *unknown*, a unary function STATUS that maps node identifiers the values *unknown* or *leader*, a static unary function VAL mapping node identifiers to numerical values, and a constant N for the number of nodes in the ring.

In an initial state we have $\text{UID}(i) = \textsc{val}(i)$, $\textsc{send}(i) = \textsc{val}(i)$, $\textsc{in}(i) = \mathit{unknown}$, and $\textsc{status}(i) = \mathit{unknown}$ for all i.

Then the database transformation algorithm can be specified as follows:

forall i **with** $0 \leq i \leq N - 1$
do par
$\quad \textsc{in}(i + 1 \mod N) := \mathit{send}(i) \parallel$
$\quad$ **if** $\textsc{in}(i) \neq \mathit{unknown} \wedge \textsc{in}(i) > \text{UID}(i)$
$\quad$ **then** $\textsc{send}(i) := \textsc{in}(i)$
$\quad$ **endif** $\parallel$
$\quad$ **if** $\textsc{in}(i) \neq \mathit{unknown} \wedge \textsc{in}(i) = \text{UID}(i)$
$\quad$ **then** $\textsc{status}(i) := \textsc{leader}$
$\quad$ **endif**
endpar enddo

Here, the bounded exploration witness T contains three access terms (UID, $\alpha(i), \mathit{true}$), $(\textsc{in}, \alpha(i), \mathit{true})$ and $(\textsc{send}, \alpha(i), \mathit{true})$, in which $\alpha(i)$ is defined as $i \geq 0 \wedge i < N$.

3.3 Synchronous Parallel Database Abstract State Machines

As for DB-ASMs we only define the rules for synchronous parallel database Abstract State Machines (parDB-ASMs) – the definition of the machines as such is the same as for DB-ASMs [7]:

- If $t_0, \ldots, t_n$ are terms over Σ, and f is a n-ary function symbol in Σ, then $f(t_1, \ldots, t_n) := t_0$ is a rule r in $\mathcal{R}$ called *assignment rule*.
- If φ is a Boolean term and $r' \in \mathcal{R}$ is a DB-ASM rule, then **if** φ **then** r' **endif** is a rule r in $\mathcal{R}$ called *conditional rule*.

oarieli@mta.ac.il

- If φ is a Boolean term with $\{x_1, \ldots, x_k\} \subseteq fr(\varphi)$ and $r' \in \mathcal{R}$ is a DB-ASM rule, then **forall** $x_1, \ldots, x_k$ **with** φ **do** r' **enddo** is a rule r in $\mathcal{R}$ called *forall rule*.
- If $r_1, \ldots, r_n$ are rules in $\mathcal{R}$, then the rule r defined as **par** $r_1 \ldots r_n$ **endpar** is a rule in $\mathcal{R}$, called *parallel rule*.
- If r_1, r_2 are rules in $\mathcal{R}$, then the rule r defined as **seq** $r_1 \;\; r_2$ **endseq** is a rule in $\mathcal{R}$, called *sequence rule*.
- If $r' \in \mathcal{R}$ is a DB-ASM rule and θ is a location function that assigns location operators ρ to terms t with $var(t) \subseteq fr(r')$, then **let** $\theta(t) = \rho$ **in** r' **endlet** is a rule $r \in \mathcal{R}$ called *let rule*.

So, the only extension is the scope of variables in the forall rules, whereas choice rules have been discarded. First, we can generalise Theorem 1.

Theorem 3. *Each parDB-ASM $\mathcal{M}$ defines a synchronous parallel database transformation t with the same signature and background as $\mathcal{M}$.*

Proof (sketch). We have to show that the four postulates for synchronous parallel database transformations are satisfied. As for the sequential time and background postulates 6 and 3, these are already built into the definition of a parDB-ASM. The same holds for the abstract state postulate 2 as far as the definition of states is concerned, and the preservation of isomorphisms is straightforward. Thus, we have to concentrate on the bounded exploration postulate 4.

For this we note that assignment rules within a parDB-ASM rule r are decisive for an update set $\Delta(r, S)$ of update sets over any state S. If $f(t_1, \ldots, t_n) := t_0$ is an assignment rule occurring within r, and $val_{S,\zeta}(t_i) = val_{S',\zeta}(t_i)$ holds for all $i = 0, \ldots, n$ and all variable assignments ζ that have to be considered, then we obtain $\Delta(r, S) = \Delta(r, S')$.

We use this to define a bounded exploration witness T. If t_i is ground, we add the access term $\alpha = t_i$ to T. If t_i is not ground, then the corresponding assignment rule must appear within the scope of a forall rule introducing the variables in t_i. Thus, variables in t_i are bound by a Boolean term φ, i.e. for $fr(t_i) = \{x_1, \ldots, x_k\}$ the relevant variable assignments are $\zeta = \{x_1 \mapsto b_1, \ldots, x_k \mapsto b_k\}$ with $val_{S,\zeta}(\varphi) = true$. Bringing φ into a form that only uses conjunction, negation and existential quantification with atoms $\beta_i = \alpha_i$ $(i = 1, \ldots, \ell)$, we can extract a set of access terms $\{(\beta_1, \alpha_1), \ldots, (\beta_\ell, \alpha_\ell)\}$ such that if S and S' coincide on these access terms, they will also coincide on the formula φ. This is possible, as we evaluate access terms by sets, so conjunction corresponds to union, existential quantification to projection, and negation to building the (finite) complement. We add all the access terms $(\beta_1, \alpha_1), \ldots, (\beta_\ell, \alpha_\ell)$ to T. □

3.4 The Equivalence Result for Parallel Database ASMs

Next we want to generalise Theorem 2.

Theorem 4. *For every synchronous parallel database transformation t there exists an equivalent parDB-ASM $\mathcal{M}$ with the same background.*

oarieli@mta.ac.il

As for the DB-ASM thesis ths theorem follows from a sequence of lemmata, which we will sketch now. We start again providing the key link from updates as implied by the state transitions to parDB-ASM rules.

Definition 3. Let T be a bounded exploration witness for the parallel database transformation t. A term that is constructed out of the subterms of $\alpha \in T$ and variables $x_1, \ldots, x_k$, for which there are access terms $(f_1, \beta_1, \alpha_1), \ldots, (f_\ell, \beta_\ell, \alpha_\ell) \in T$ such that $\bigcup_{i=1}^{\ell} fr(\beta_i) \cup fr(\alpha_i) = \{x_1, \ldots, x_k\}$ holds is called a *critical term*.

If γ is a critical term, let $(f_1, \beta_1, \alpha_1), \ldots, (f_\ell, \beta_\ell, \alpha_\ell)$ be the access terms used in its definition. For a state S choose $b_1, \ldots, b_k \in B$ with $val_{S,\zeta}(\beta_i) = val_{S,\zeta}(\alpha_i)$ with $\zeta = \{x_1 \mapsto b_1, \ldots, x_k \mapsto b_k\}$ for $i = 1, \ldots, \ell$, and let $a = val_{S,\{x_1 \mapsto b_1, \ldots, x_k \mapsto b_k\}}(\gamma)$.

Definition 4. For a state S of a database transformation t let $C_S = \{val_S(\alpha) \mid \alpha \in T\} \cup \{true, false, \bot\}$ and $B_S = \{a_i \mid f(a_1, \ldots, a_n) \in val_S(\beta, \alpha)$ for some access term $(f, \beta, \alpha) \in T\}$. Then $\bar{C}_S$ is the *background closure* of $C_S \cup B_S$ containing all complex values that can be constructed out of $C_S \cup B_S$ using the constructors and function symbols (interpreted in S) in the background. The elements of $\bar{C}_S$ are called the *critical elements* of S.

The following lemma and its proof are analogous to the result in [6, Lemma 6.2].

Lemma 1. *For all updates $(f(a_1, \ldots, a_n), a_0) \in \Delta(t, S)$ the values $a_0, \ldots, a_n$ are critical elements of S.*

In [6] it is a straighforward consequence of Lemma 6.2 that individual updates can be represented by assignments rules, and consistent update sets by par-blocks of assignment rules. In our case showing that $\Delta(t, S)$ can be represented by a parDB-ASM rule requires a bit more work. We address this in the next lemma.

Lemma 2. *Let t be a database transformation. For every state $S \in \mathcal{S}_t$ there exists a rule r_S such that $\Delta(t, S) = \Delta(r_S, S)$, and r_S only uses critical terms.*

Proof (sketch). Consider any update $u = (f(a_1, \ldots, a_n), a_0) \in \Delta(t, S)$. According to Lemma 1 the values $a_0, \ldots, a_n$ are critical and hence representable by terms involving variables from access terms in T, i.e. $a_i = val_{S,\zeta}(t_i)$ with either $fr(t_i) \subseteq \{x_1, \ldots, x_k\}, \zeta = \{x_1 \mapsto b_1, \ldots, x_k \mapsto b_k\}$ and

$$(b_1, \ldots, b_k) \in \mathcal{B}_u = \{(b_1, \ldots, b_k) \in B^k \mid \bigwedge_{1 \le i \le \ell} val_{S,\zeta}(\beta_i) = val_{S,\zeta}(\alpha_i)\}$$

with access terms $(f, \beta_i, \alpha_i) \in T$ $(i = 1, \ldots, \ell)$ and $fr(\beta_i) \subseteq \{x_1, \ldots, x_k\}$, or t_i is a ground critical term.

As the case that all terms $t_0, \ldots, t_n$ are ground terms is straightforward, let us concentrate on the case that at least one of the terms $t_0, \ldots, t_n$ is not a ground

oarieli@mta.ac.il

term. We first assume that none of terms $t_0, \ldots, t_n$ contain location operators. The access terms (β_i, α_i) define a finite set of locations

$$\begin{aligned} L \quad = \quad & \{f(a_1, \ldots, a_n) \mid a_i = val_{S,\zeta}(t_i) \text{ for } i = 1, \ldots, n, \text{ and} \\ & \qquad \zeta = \{x_1 \mapsto b_1, \ldots, x_k \mapsto b_k\} \text{ for } (b_1, \ldots, b_k) \in \mathcal{B}_u\}. \end{aligned}$$

We may assume that the set $\mathcal{B}_u$ is minimal in the sense that we may not find additional access terms that would define a subset $\mathcal{B}'_u \subsetneq \mathcal{B}_u$ still containing the value tuple $(b_1, \ldots, b_k)$ that is needed to define the update u. If ℓ is updated in S, then (ℓ, a_0) is represented by the assignment rule $f(t_1, \ldots, t_n) := t_0$ with $x_1, \ldots, x_k$ interpreted by $(b_1, \ldots, b_k) \in \mathcal{B}_u$.

Next we revise the previous assumption that none of terms $t_0, \ldots, t_n$ contain location operators to a general case, i.e. location operators may appear in the terms $t_0, ..., t_n$ of an assignment rule $f(t_1, ..., t_n) := t_0$. Let f_ℓ be a unary function symbol such that $x_{t_i} = f_\ell(i)$, then, without loss of generality, we can replace the terms $t_1, ..., t_n$ of an assignment rule $f(t_1, ..., t_n) := t_0$ with the variables $x_{t_1}, ..., x_{t_n}$, such that

```
seq
    par
        x_{t_1} := t_1
            ⋮
        x_{t_n} := t_n
    endpar
    f(x_{t_1}, ..., x_{t_n}) := t_0
endseq
```

It means that we can simplify the construction of rules for updates which may correspond to terms with location operators by only considering the case that location operators appear at the right hand side of an assignment rule. If a term t_i $(i \in [1, n])$ at the left hand side contains a location operator, by the above translation, we may treat it as being a term at the right hand side of another assignment rule again.

Suppose that the outermost function symbol of term t_0 is a location operator ρ, e.g. $t_0 = \rho(m)$ where $m = \langle t_0^{'}|$ for all values $\overline{a} = (a_1, ..., a_p)$ in $\overline{y} = (y_1, ..., y_p)$ such that $val_{S,\zeta[x_1 \mapsto b_1, \ldots, x_k \mapsto b_k]}(\varphi(\overline{x}, \overline{y})) = true\rangle$, and $\overline{x}$ denotes a tuple of variables among $x_1, ..., x_k$. Then for each assignment rule $f(x_{t_1}, ..., x_{t_n}) := t_0$ in which t_0 contains a location operator as described before, we can construct the following rule to remove the location operator ρ by a let rule and a forall rule:

```
let θ(f(x_{t_1}, ..., x_{t_n})) = ρ in
    forall y_1,...,y_p with φ(x̄, ȳ)
    do
        f(x_{t_1}, ..., x_{t_n}) := t_0'
    enddo;
endlet
```

oarieli@mta.ac.il

This construction can be conducted iteratively. If the outermost function symbol of the above term $t_0^{'}$ is a location operator, then we need to construct a rule in a similar way to replace the assignment rule $f(x_{t_1}, ..., x_{t_n}) := t_0^{'}$. This procedure continues until the right hand side of an assignment rule is a term without any location operator. □

With Lemma 2 we have done the hard part of the proof. The following lemmata are almost identical to those in the proof of the DB-ASM thesis [7].

Lemma 3. *Let $S, S' \in \mathcal{S}_t$ be states that coincide on the set CT of critical terms. Then $\Delta(r_S, S') = \Delta(t, S')$ holds.*

Proof. As S and S' coincide on CT, they also coincide on T, which gives $\Delta(t, S) = \Delta(t, S')$ by the bounded exploration postulate. Furthermore, we have $\Delta(r_S, S) = \Delta(t, S)$ by Lemma 2. As r_S uses only critical terms, the updates produced in state S must be the same as those produced in state S', i.e. $\Delta(r_S, S) = \Delta(r_S, S')$, which proves the lemma.

Lemma 4. *Let S, S_1, S_2 be states with S_1 isomorphic to S_2 and $\Delta(r_S, S_2) = \Delta(t, S_2)$. Then also $\Delta(r_S, S_1) = \Delta(t, S_1)$ holds.*

Proof. Let σ be an isomorphism from S_1 to S_2. Then $\Delta(r_S, S_2) = \sigma(\Delta(r_S, S_1))$ holds, and the same applies to $\Delta(t, S_2) = \sigma(\Delta(t, S_1))$. As we presume $\Delta(r_S, S_2) = \Delta(t, S_2)$, we obtain $\sigma(\Delta(r_S, S_1)) = \sigma(\Delta(t, S_1))$ and hence $\Delta(r_S, S_1) = \Delta(t, S_1)$, as σ is an isomorphism.

Next, in the spirit of [6] we want to extend the equality of sets of update sets for t and r_S to a larger class of states by exploiting the finiteness of the bounded exploration witness T. For this we define the notion of T-equivalence similar to the corresponding notion for the sequential ASM thesis, with the difference that in our case we cannot take T, but must base our definition and the following lemma on CT.

Definition 5. States $S, S' \in \mathcal{S}_t$ are called *T-similar* iff $E_S = E_{S'}$ holds, where E_S is an equivalence relation on CT defined by

$$E_S(\gamma_1, \gamma_2) \;\Leftrightarrow\; val_S(\gamma_1) = val_S(\gamma_2).$$

Lemma 5. *We have $\Delta(r_S, S') = \Delta(t, S')$ for every state S' that is T-similar to S.*

Proof. Replace every element in S' that also belongs to S by a fresh element. This defines a structure S_1 isomorphic to S' and disjoint from S. By the abstract state postulate S_1 is a state of t. Furthermore, by construction S_1 is also T-similar to S' and hence also to S.

Now define a structure S_2 isomorphic to S_1 such that $val_{S_2}(\gamma) = val_S(\gamma)$ holds for all critical terms $\gamma \in CT$. This is possible, as S and S_1 are T-similar, i.e. we have $val_S(\gamma_1) = val_S(\gamma_2)$ iff $val_{S_1}(\gamma_1) = val_{S_1}(\gamma_2)$ for all critical terms γ_1, γ_2. By the abstract state postulate S_2 is also a state of t.

Using Lemma 3 we conclude $\Delta(r_S, S_2) = \Delta(t, S_2)$, and by Lemma 4 we obtain $\Delta(r_S, S') = \Delta(t, S')$ as claimed.

oarieli@mta.ac.il

We are now able to prove our main result, first generalising Lemma 2 to multiple-steps updates in the next lemma, from which the proof of the main characterisation theorem is straightforward.

Lemma 6 (sketch). *Let t be a synchronous parallel database transformation with signature Σ. Then there exists a parDB-ASM rule r over Σ, with same background as t such that $\Delta(r, S) = \Delta(t, S)$ holds for all states $S \in \mathcal{S}_t$.*

Proof. In order to decide whether equivalence relations E_S and $E_{S'}$ coincide for states $S, S' \in \mathcal{S}_t$ it is sufficient to consider the subset $CT' \subseteq CT$ defined by the bounded exploration witness T as in Definition 3. Hence, as T is finite, CT' is also finite, and consequently there can only be finitely many such equivalence relations. Let these be $E_{S_1}, \ldots, E_{S_n}$ for states $S_1, \ldots, S_n \in \mathcal{S}_t$.

For $i = 1, \ldots, n$ construct Boolean terms φ_i such that $val_S(\varphi_i) = true$ holds iff S is T-similar to S_i. For this let $CT' = \{\gamma_1, \ldots, \gamma_m\}$, and define terms

$$\bar{\gamma}_j = \begin{cases} \gamma_j & \text{if } \gamma_j \text{ is closed} \\ \langle (x_1, \ldots, x_k) \mid \bigwedge_{1 \le i \le \ell} \beta_i = \alpha_i \rangle & \text{if } \gamma_j = (x_1, \ldots, x_k) \text{ with variables taken} \\ & \text{from } (\beta_1, \alpha_1), \ldots, (\beta_\ell, \alpha_\ell) \end{cases}$$

exploiting the fact that the background structures provide constructors for multisets and pairs (and thus also tuples). Then

$$\varphi_i = \bigwedge_{\substack{1 \le j_1, j_2 \le m \\ E_{S_i}(\gamma_{j_1}, \gamma_{j_2})}} \bar{\gamma}_{j_1} = \bar{\gamma}_{j_2} \quad \wedge \bigwedge_{\substack{1 \le j_1, j_2 \le m \\ \neg E_{S_i}(\gamma_{j_1}, \gamma_{j_2})}} \bar{\gamma}_{j_1} \neq \bar{\gamma}_{j_2}$$

asserts that $E_S = E_{S_i}$ holds. Now define the rule r by

par **if** φ_1 **then** r_{S_1} **endif**
if φ_2 **then** r_{S_2} **endif**
$\vdots$
if φ_n **then** r_{S_n} **endif endpar**

If $S \in \mathcal{S}_t$ is any state of t, then S is T-equivalent to exactly one S_i $(1 \le i \le n)$, which implies $val_S(\varphi_j) = true$ iff $j = i$, and hence $\Delta(r, S) = \Delta(r_{S_i}, S) = \Delta(t, S)$ by Lemma 5.

This completes the proof of Theorem 4.

4 Conclusion

In this paper we generalise the DB-ASM thesis [7], i.e. the exact characterisation of database transformations by a variant of Abstract State Machines, to synchronous parallel database transformations. The key ideas in our previous definition of database transformations were

oarieli@mta.ac.il

- to capture the intrinsic finiteness of databases by adopting meta-finite structures [4] for the states,
- to capture the constructs for the data model by means of background structures,
- to capture aggregation in queries by particular multiset functions, and
- to permit special non-ground terms in bounded exploration witnesses to capture unbounded parallelism on the database part of a state.

The generalisation in this paper simply exploits the observation that the restriction of the extended non-ground terms in bounded exploration witnesses to the database part of the state signature is not necessary, while the other postulates remain unchanged.

As a consequence of the research presented in this paper we may first ask what what happens if we drop the request that states are meta-finite structures, and use simply any Tarski structures. This would mean to look only at the algorithmic part of the state. We would obtain a different thesis for synchronous parallel algorithms. The question to be investigated is how this thesis would differ from Blass' and Gurevich's parallel ASM thesis in [2], which is still debated in the ASM community. We conjecture that our result only provides a simplification of the postulates. Whether it is more intuitive has to be explored.

Naturally, synchronous parallel database transformations can be tailored to particular data models, as this only requires to adapt the background structures. For the case of XML the corresponding DB-ASMs have been investigated in detail in [8]. Furthermore, the complete logic for DB-ASMs (see [10] for a first version) should carry over to parallel DB-ASMs.

However, the DB-ASM thesis in [7] also supports bounded non-determinism, which for the sake of keeping the presentation simple was not handled in this paper. It should be possible to preserve the bounded non-determinism also for synchronous parallel database transformations, but the corresponding (more difficult) proofs have to be checked carefully.

With respect to the complete logic for DB-ASMs, however, we doubt that a generalisation would be straightforward (if possible at all), as our theory relies on the boundedness arising from the finiteness of the database part. Nonetheless it will be worth some research effort.

Finally, it appears to be much more challenging to approach also asynchronous parallel database transformations. We believe that in this case (different from some attempts by Yuri Gurevich) we cannot assume to have global states, so non-determinism becomes essential. We plan to investigate into this research direction.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley Longman Publishing Co., Inc., Boston (1995)
2. Blass, A., Gurevich, Y.: Abstract state machines capture parallel algorithms. ACM Transactions on Computational Logic 4(4), 578–651 (2003)

oarieli@mta.ac.il

3. Börger, E., Stärk, R.: Abstract State Machines. Springer, Heidelberg (2003)
4. Grädel, E., Gurevich, Y.: Metafinite model theory. Information and Computation 140(1) (1998)
5. Gurevich, Y., Tillmann, N.: Partial updates. Theoretical Computer Science 336(2-3), 311–342 (2005)
6. Gurevich, Y.: Sequential abstract state machines capture sequential algorithms. ACM Transactions on Computational Logic 1(1), 77–111 (2000)
7. Schewe, K.-D., Wang, Q.: A customised ASM thesis for database transformations. Acta Cybernetica 19(4), 765–805 (2010)
8. Schewe, K.-D., Wang, Q.: XML database transformations. Journal of Universal Computer Science 16(20), 3043–3072 (2010)
9. Schewe, K.-D., Wang, Q.: Partial updates in complex-value databases. In: Heimbürger, A., et al. (eds.) Information and Knowledge Bases XXII. Frontiers in Artificial Intelligence and Applications, vol. 225, pp. 37–56. IOS Press (2011)
10. Wang, Q.: A Proof System with Bounded Non-determinism in Database Transformations. In: Schewe, K.D., Thalheim, B. (eds.) SDKB 2010. LNCS, vol. 6834, pp. 114–133. Springer, Heidelberg (2011)

oarieli@mta.ac.il

Functional Dependencies on Extended Relations Defined by Regular Languages*

Gyula I. Szabó and András Benczúr

Eötvös Loránd University, Faculty of Informatics, Hungary
1118, Budapest, Pázmány Péter sétány, 1/C
gyula@szaboo.de, abenczur@inf.elte.hu

Abstract. In this paper, we first rephrase the notion of extended tuple as a sentence from a regular language generated by a grammar G where the nonterminal symbols of the grammar are the attribute names of the tuple. Finite sets of extended tuples play the role of extended relation instances. Then we introduce the dual language, which generates the accepted tuple-types of the extended relation. We define the syntactical form of functional dependencies for extended relation on the graph of the finite state automaton associated to the dual language. Using this model we can handle extended relations generated by recursive regular expressions too. The implication problem of our class of dependencies is decidable by a version of Chase algorithm specified on the graph of the associated FSA.

1 Introduction

XML has recently been practically the standard format of data exchange over the world wide web. XML was originally defined for describing and presenting individual documents, but it has been used for building databases, and, especially, accessing relational data bases through XML views as well([2]). Because of these uses of XML, integrity constraints play a role of increasing importance in the XML world. Preserving integrity constraints during data base transformation is a useful method to control the equivalence of the transformed data base with the original one (relational to XML or vice versa) ([5], [7], [11]). When accessing relational data bases through XML views, especially when updating data, complying with integrity constraints secures the consistency of the data base. Constraints are important in planning and designing XML documents, they help to create a consistent data model.

Functional dependency (FD) is perhaps the most important integrity constraint for any data model, but surely, it is the most exhaustively analyzed one of them. In relational data model, FDs are well studied and are widely used in normalization theory([1]). XML functional dependency has been defined on some different ways but no general accepted definition was reached until now.

* The research was supported by the European Union and co-financed by the European Social Fund (grant agreement no. TAMOP 4.2.1./B-09/1/KMR-2010-0003).

T. Lukasiewicz and A. Sali (Eds.): FoIKS 2012, LNCS 7153, pp. 384–403, 2012.
© Springer-Verlag Berlin Heidelberg 2012

oarieli@mta.ac.il

The main problem with defining functional dependency in the XML context is the lacking "tuple" concept for XML. An instance of a relational schema is a set of tuples, and one can easily select pairs of tuples from this set for comparing in order to check whether the instance satisfies a given functional dependency defined on the relational schema. In the XML world there is no general accepted definition for the concept of tuple, and even if one chooses a collection of elements and declares them to be a "tuple", it is very hard to find a proper matching algorithm for them. Arenas and Libkin defined "tree tuples" in their seminal work [3], based upon DTD schema. Vincent et al.[14] described some cases, not covered with "tree tuples", and invented the notion "closest node" to deal with them. They defined functional dependency on XML trees without any schema, and used DTDs just to prove that their definition is equivalent with "tree tuples" for some classes of DTDs. Hartmann at al. ([10]) define a special form of functional dependency with properties based upon tree homomorphism. They use XML Schema to describe the dependency.

Wang in 2005 [15] compared a lot of different functional dependency definitions for XML and proposed a new definition of XFD which unifies and generalizes the surveyed XFDs: all these XFD definitions base upon path expressions created from DTDs or XML Schema definitions. Since then some new definitions were presented. Lately, Hartmann at al. in another publication ([8]) surveyed XML functional dependencies in order to preparing the study of Armstrong data trees: they focused on structural (parent-child) properties.

All of these FD concepts are very intricate, compared with the classical functional dependency concept for relational databases. In the case of XML data model they base mostly upon path expressions. Our main motivation was to find a simple, but general definition for functional dependency for a broad family of data models: our only assumption is that the "tuples" should be sentences of a given regular language (i.e., they should be described by a regular expression).

We generalize the concept of functional dependencies defined on attributes of a relational schema in a natural way to the sentences of the dual language associated to the "tuple"-generator regular language.

Our functional dependency concept is principally not a new member of the broad XFD (XML FD) family, it is defined on general regular languages, but it can be applied for XML too, as long as the element declaring expressions used by the XML schema definition comply with our assumptions. From our point of view, an XML document is a set of text fragments, each one of them is a string of symbols, and some of those strings have type as sentences of a regular language. We focus in our approach on the instances of a given type spread over the XML document. So in our investigations the regular tree model of XML ([6]) has no importance.

Regular path languages play an important role in dealing with XML keys ([9],[4]). Our concept differs from theirs in that we consider contents of siblings in place of parent-child relations.

oarieli@mta.ac.il

2 Extended Relations for Regular Types

The ELEMENT! declaration in DTD and complex type definition in XML Schema is given by a regular expression on the set of component (tag) names. This regular expression generates a regular language with component names as (terminal) symbols (*tag* language). The grammar of this language is usually recursive because of the Kleene-star. Instances of the DTD element declarations or the XSD complex types are elements of XML documents that are satisfying the given DTD or XML Schema. The text values of these element instances are also sentences of a regular language (*content* language) defined by a regular grammar, with tag-names as non-terminals and element text values as terminals. Starting symbol is the tag-name of the declared element and we use a common closing symbol too. The element instances themselves can be seen as traversing on the FSA for the *content* language: the visited nodes (states) are the tags, the visited edges (values) are the element contents (text values).

We consider the XML element declaration as a structure, that contains immediate subelements only. All subelements have their own domain (either PCDATA for DTD, atomic type for XML Schema, or, any complex value with not examined internal structure).

Starting from a general regular language (from the *content* language in the XML case), for each sentence we can view the sequence of non-terminals used during the derivation of the sentence as another sentence. We call this sentence, built up from non-terminals, the dual sentence, and the associated language is the dual language (*tag* language in the XML case). Symbols in the dual sentences play the same role as attribute names in the relational model, and dual sentences play the role of row-type constructors.

We introduce a couple of notions to prepare the general definition of functional dependency on regular languages. Let G(N,T,S,P) be a regular grammar, denote L(G) the generated language. Let us define the dual language of G as follows: Let $\omega \in L(G)$ be a sentence that was generated by the substitutions $S \Rightarrow d_0\, N_1, N_1 \Rightarrow d_1\, N_2, \ldots, N_k \Rightarrow d_k$, short by $Sd_0N_1 \ldots N_kd_k$ (We use the notation "$\Rightarrow$" for the symbol substitution in the production rules to avoid confusion with "$\rightarrow$", that we use to denote functional dependency).

The nonterminal symbols used in the substitutions build up the w dual (associated) sentence for ω. To this fixed dual sentence more than one sentence of the original language L(G) can be related to, we say that these sentences are the tuples related to the w dual sentence as extended relation. The type of these tuples is $< S : D_0, N_1 : D_1, \ldots, N_k : D_k >$, where D_i is the domain of N_i, analogously to the relational data model.

The dual sentences associated to G set up the dual language of G. The dual language of a regular language is also regular, its G'(N',N,S',P') grammar follows from G easily.

oarieli@mta.ac.il

We can represent the dual language of L(G) by the corresponding finite state automaton M. Each path on the graph of M that is rooted in S and ends in an accepting state represents a dual sentence.

Each sentence of the language L(G') specifies a type for tuples, these types set up the schema of our "extended relation". An instance of this relation , R, is a finite set of tuples (sentences of L(G)), so that each sentence will be extended with its associated dual sentence.

The type of a tuple $t \in R$ is a sentence of L(G'), that is, R is the set of tuples associated with (possibly different) dual sentences as types.

Let us see a relational example.

Example 1. Let R(A,B,C,D,E) be a relational schema with

A,B,C,D,E as the set of attributes,
dom as the domain of the values for the attributes.

Let G(N,T,S,P) be a grammar, where

$N = \{R, A, B, C, D, E, \mathcal{END}\}$ is the set of nonterminals (we may associate different domains to each attribute),
$T = \{\langle, \rangle, dom\}$ is the set of terminal values,
$S = \{R\}$ is the start symbol,
$P = \{R \Rightarrow \langle A, A \Rightarrow aB, B \Rightarrow bC, C \Rightarrow cD, D \Rightarrow dE, E \Rightarrow e\,\mathcal{END}, \mathcal{END} \Rightarrow \rangle\}$;
$a, b \ldots \in dom$, are the production rules.

The L(G) language, generated by G, consists of strings "abcde" (tuples), concatenated values from ***dom***, and the instances of R are finite subsets of L(G).

The following XML example is based upon a slightly more complicated regular grammar as above, derived from a regular expression.

Example 2. The XML document on Fig. 1 conforms to the DTD on Fig. 2.

The ELEMENT! declaration *book* in Fig. 2, as a regular expression defines a regular grammar G(N,T,S,P), where

$N = \{book, title, author, publisher, \mathcal{END}\}$
$T = \{\langle, \rangle, d_1, d_2, \ldots\}$
$S = \{book\}$
$P = \{book \Rightarrow \langle\, title, title \Rightarrow d_1\, author, author \Rightarrow d_2\, author)$
$\{author \Rightarrow d_2\, publisher, publisher \Rightarrow d_3\, \mathcal{END}, \mathcal{END} \Rightarrow \rangle\}$
$d_1, d_2 \ldots \in dom$

oarieli@mta.ac.il

```
<book>
<title>Foundations of Databases</title>
<author>S. Abiteboul</author>
<author>R. Hull</author>
<author>V. Vianu</author>
<publisher>AddisonWesley</publisher>
</book>
```

Fig. 1. Example XML document

```
<!DOCTYPE book [
<!ELEMENT book (title, author+, publisher)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
]>
```

Fig. 2. An example DTD for the XML document of Fig. 1

The corresponding dual grammar, according to our concept of dual languages let be G'(N',N,S',P'), where

$N' = \{Book, Title, Author, Publisher, \mathcal{END}\}$
$N = \{\langle, \rangle, title, author, publisher\}$
$S' = \{Book\}$
$P' = \{Book \Rightarrow \langle Title, Title \Rightarrow title\, Author, Author \Rightarrow author\, Author)$
$\{Author \Rightarrow author\, Publisher, Publisher \Rightarrow publisher\, \mathcal{END}, \mathcal{END} \Rightarrow \rangle\}$

The language L(G) generates a set of XML text fragments, in turn, the dual language L(G') consists of strings of XML element names (tags).

3 FD on Regular Languages

3.1 Functional Dependency Defined on the Dual Language

According to the concept of relational functional dependency we can select two subsequences on each dual sentence, as the left and right side of the dependency, let we state this as the syntactic specification for the dependency. The set of sentences R satisfies this dependency, when there exist no two tuples in R so, that they are identical in all the nonterminals fitting to the left side subsequences, but on the subsequences selected for the right side, they differ on at least one position.

Notice, that we specify the syntax of the dependency on the dual language, its satisfaction will be checked on the accepted sentences of the language L(G).

Let G=(N,T,S,P) be a regular grammar. We can represent the dual language of L(G) by the corresponding finite state automaton M. We denote the graph representation of the automaton for G by M(G)(it is rooted, directed and simple). The nodes of M(G) are the symbols of the set N, and there is a special node,

oarieli@mta.ac.il

$\mathcal{START}$ assigned to the initial state, another special node, $\mathcal{END}$ assigned to the end (final) state. The edges of M(G) comply with the production rules P: the $A \Rightarrow xB$ rule means that from the node A goes a directed edge to the node B. The $A \Rightarrow x$ rule means that from the node A goes a directed edge to the node $\mathcal{END}$. Each traversing on the graph M(G) rooted in $\mathcal{START}$ and ending in $\mathcal{END}$ represents a dual sentence. The dual sentences build up a traversal of M(G) (we use the notion *traversing* for the visited nodes in the sequence of visiting during a walk on a directed path, a *traversal* visits all nodes of the graph).

If the grammar G is non-recursive then the associated L(G') dual language is finite, because there is only a finite number of paths in the graph M(G) (the graph representation of the finite state automaton for L(G)). We can define functional dependencies (left and right side) on each one of these dual sentences, we can check the logical implications among them.

If the grammar G is recursive then the associated L(G') dual language is infinite. We can use the pumping process to select substrings of dual sentences for defining functional dependencies on them. We can select sub-paths on the non-pumped area as described before,(pumping 0-times the circles), then on the pumped part we can select left and right side of the dependency and pumping them together with the same frequencies. It is important, that we should always consider the visited nodes for a whole path (from $\mathcal{START}$ to $\mathcal{END}$) in the sequence of the processing, and the repeatings (recursions) should also be taken in sequence. For all dual sentences should be the selection unique for the left and right sides of the dependency.

In order to specify the (left and right) sides of a functional dependency we should pick up two sets of nodes from the graph M(G): one set for the left side (denoted by X), another one for the right side (denoted by Y). We can choose nodes visited by a traversing and state that each visiting of these nodes would be selected. We can choose starting and ending points for a path in the traversing, so that this pair of nodes will be selected at each closing of that path. We can specify the selection by a tuple: Y=(Y_1,Y_2), where $Y_1 \subseteq N$ and Y_2 is a subgraph of the transitive closure of M(G). We refer to this tuple as an assignment. Y_1 is taken from the non-recurred part of M(G), Y_2 refers to nodes and edges whose are (could be) repeatedly visited during the traversing.

Y selects a unique subsequence from a given dual sentence after either of two methods as follows:

Let $w = \{v_1\, v_2\, \ldots\, v_n\}$ be a dual sentence over M(G)=(V,E), and denote $walk\,(w) = \{\mathcal{START}, v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n, \mathcal{END}\}$ (short by $walk\,(w) = \{v_1, v_2, \ldots, v_n\}$) the corresponding traversing on M(G).

Definition 1 (Strict Selection). *Let $Y = (Y_1, Y_2)$ be an assignment and let w be a dual sentence over M(G). Let $walk\,(w) = \{v_1, v_2, \ldots, v_n\}$. The symbols in Y_1 will be selected in order of their exploration (when visited). For each edge $e \in Y_2$ when the edge will be closed on the shortest path between its endpoints during the traversing on $walk\,(w)$, these two endpoints will be selected in their succession order (when visited at all). That is, if the two endpoints of the closing*

oarieli@mta.ac.il

path are A and B ($A = v_i, B = v_j$ for some $1 \leq i < j \leq n$) then that path will be selected which does not contain neither A nor B. The nodes in Y_2 will be selected by each visiting (if any) during the traversing on walk (w). The selection will be processed for all edges and nodes in Y_2 autonomously. By the end of the selection the from w selected symbols build up the (possibly empty) array $w[Y] = \{v_{i_1}, \ldots, v_{i_k}\}$ ($1 \leq i_1 < i_2 < \ldots < i_k \leq n\,(k \geq 0)$).

Let t∈L(G), the corresponding dual sentence w ∈ L(G'). We can interpret the w[Y] sequence of symbols as set of "attributes", that projects the "tuple" t to the values t[Y]. If w[Y]={}, then t[Y]={} as well.

Definition 2 (Full Selection). *Let $Y = (Y_1, Y_2)$ be an assignment and let w be a dual sentence over M(G). Let walk $(w) = \{v_1, v_2, \ldots, v_n\}$. For the edge $(A, B) \in Y_2$ the A node will be selected at each visiting, when it will be followed with visiting B during the traversing on walk (D), and the node B will be selected at each visiting, when it follows the node B (when visited at all). The edge (A,B) does not induce the selection of node A from the traversing, when the node B is not present among its followers, and the selection of B will be avoided, when there is no prior visited A node. The full selection means that between the firstly traversed node A and the lastly traversed node B all A and B nodes will be selected. The nodes in Y_2 will be selected by each visiting during the traversing on walk (w). The selection will be processed for all edges and nodes in Y_2 autonomously. By the end of the selection the selected symbols from w build up the (possibly empty) array $w\{Y\} = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ ($1 \leq i_1 < i_2 < \ldots < i_k \leq n\,(k \geq 0)$).*

Let t∈L(G), the corresponding dual sentence w ∈ L(G'). We can interpret the w{Y} sequence of symbols as set of "attributes", that projects the "tuple" t to the values t{Y}. If w{Y}={}, then t{Y}={} as well.

Example 3. Let the edge $(A, B) \in Y_2$. Let a traversing in M(G) be

BCADCACAEBDCBAEB.

The strict selection selects the non-capitalized nodes enumerated in the subscript according to the sequence numbers of the visiting:

$$BCADCACa_1Eb_1DCBa_2Eb_2.$$

The full selection selects the nodes of the edge (A,B) each time it is visited, the selected non-capitalized nodes are enumerated in the subscript according to the sequence numbers of the (recursive) visiting:

$$BCa_1DCa_2Ca_3Eb_1DCb_2a_4Eb_3.$$

The strict and full selections offer different methods for including recursive parts of regular expressions to be considered in functional dependencies. Built upon them, we have two more possible, autonomous variations for the syntactic of the dependencies (we use the notation of the strict selection in the following, but the full selection could be used as well):

oarieli@mta.ac.il

Definition 3 (Variant A: Two-steps Selection). *Let w be a dual sentence and let* $X = (X_1, X_2)$ *and* $Y = (Y_1, Y_2)$ *be two assignments over M(G), so, that* $X_1 \subseteq Y_1$ *and* $X_2 \subseteq \overline{Y}_2$ *(where* $\overline{Y}_2$ *denote the transitive closure of* Y_2*). The selection process that creates* $w[Y]$ *visits the nodes and edges of X at the same time: we denote the resulting selection by* $w[Y[X]]$.

Let t∈L(G), the corresponding dual sentence w ∈ L(G'). We can interpret the w[Y[X]] sequence of symbols as set of "attributes", that projects the "tuple" t to the values t[Y[X]].

Definition 4 (Variant B: One-step or Simple Selection). *Let w be a dual sentence and let* $X = (X_1, X_2)$ *and* $Y = (Y_1, Y_2)$ *be two assignments over M(G). We use one-step selection for X and Y, when the selections* $w[X]$ *and* $w[Y]$ *will be performed autonomously.*

Example 4. Let us see the difference between the simple and two-steps (w[Y[X]]) selection. Figure 3 shows the graph of nodes A,B,C,D,E,F and edges (A,B), (B,C), (C,D), (D,A), (A,E),(E,F) and (F,D).

Let $Y = (\{\}, \{(A,B), (B,C), (C,D)\})$ and let $X = (\{\}, \{(A,D)\})$ be two assignments and let w=ABCDABCDAEFDABCD be a dual sentence.

Let us denote the nodes, selected by w[Y], w[Y[X]] and w[X],respectively, with non-capitals:

```
          1234567890123456
w[Y]=     abcdabcdAEFDabcd
w[Y[X]]= aBCdaBCdAEFDaBCd
w[X]=     aBCdaBCdaEFdaBCd
```

The selected nodes for a given position are different: w[X] selects for the traversing-position 9 the node A (denoted by a), but w[Y[X]] makes for the position 9 no selection (denoted by A). The reason of the different selections is the fact, that the graph Y_2 does not contain the edges (A,E),(E,F) and (F,D), so the nodes on the edge (A,D) in its transitive closure could be selected only then, when visiting them over the edges (A,B), (B,C), (C,D). With the non-capital notation this means that we can only connect the two nodes of an edge from X when walking on the nodes denoted by lower case letters in w[Y].

Definition 5 (Regular Functional Dependency). *Let L(G) be regular language and let M(G) be the graph representation of the finite state automaton for L(G). Let* $X = (X_1, X_2)$ *and* $Y = (Y_1, Y_2)$ *be two assignments over M(G) so, that* $X_1 \subseteq Y_1$ *and* $V(X_2) \subseteq V(Y_2)$*. A functional dependency defined over M(G) (regular FD) is an expression of the form* $X \rightarrow Y$*. The R (finite) database instance over L(G) satisfies the* $X \rightarrow Y$ *functional dependency (denoted by* $R \models X \rightarrow Y$*), if for any two* $t_1, t_2 \in R$ *tuples* $t_1[X]=t_2[X]$ *(*$t_1[Y[X]]=t_2[Y[X]]$ *with selection specified after the Variant A) can be fulfilled only then, when* $t_1[Y]=t_2[Y]$ *also comes true. We call the case Y=M(G) key dependency.*

oarieli@mta.ac.il

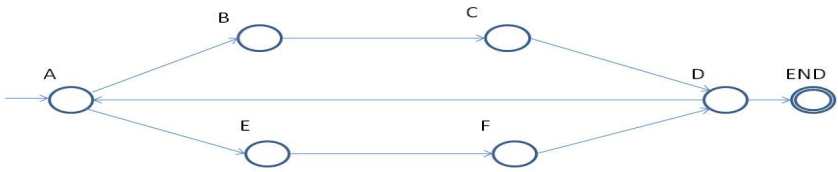


Fig. 3. Example FSA graph

Remark 1. In order to make the examples for functional dependencies better readable, we do not apply the containment assumption above ($X_1 \subseteq Y_1$ and $V(X_2) \subseteq V(Y_2)$) to them.

Example 5. Let R(A,B,C,D,E) be a relational schema and G(N,T,S,P) be a grammar as described in Example 1, and let $X = (\{B, C\}, \{\})$ and $Y = (\{D\}, \{\})$ specified after the Definition 5. For the relational case the selection methods are identical, because the graph of the corresponding finite state automaton has no circle. That is, the FD $X \to Y$ defined after Definition 5 is identical with the traditional relational functional dependency of $\{B, C\} \to \{D\}$.

An R $\in$ L(G) instance satisfies the $X \to Y$ functional dependency, when for any two generated sentences whose terminal symbols, resulted from the substitution of the nonterminal symbol B, and from the substitution of the nonterminal symbol C are equal in the two sentences, then those, that arose from the substitution of D, are also equal. The instance R collects those sentences of the language that were generated (instantiated) in a single process, at the "same time". The dependency constraint applies to this instance, and not to the whole language. This condition specified on the appropriate finite state automaton (FSA) means, that the edges chosen for the path that ends in vertex B and for the path that ends in vertex C,respectively, would determine the edge chosen for the path that ends in vertex D.

Remark 2. Throughout the following discussions, we assume that a regular language L(G) has always an associated selection method for functional dependencies chosen from the Definitions 1 - 2 and from the Definitions 3 - 4, respectively. When this associated selection method applies the Definition 3 (two-phase selection) then for a FD the two assignments (X and Y) should fulfill $X_1 \subseteq Y_1$ and $X_2 \subseteq \overline{Y}_2$ (where $\overline{Y}_2$ denote the transitive closure of Y_2).

Definition 6. *Let L(G) be regular language and let M(G) be the graph representation of the finite state automaton for L(G). Let Σ be a set of FDs and let σ be an FD over M(G), then Σ implies σ (denoted by $\Sigma \models \sigma$) when for all $R \subseteq L(G)$ (finite) database instances that satisfy Σ, $R \models \sigma$ will also be fulfilled.*

oarieli@mta.ac.il

Algorithm 1. *Algorithm for checking implication.*
Input: graph M(G)=(V,E), a set Σ and $\sigma : X \rightarrow Y$ (where X=(X_1, X_2) and Y=(Y_1, Y_2)) functional dependencies over M(G)
Output: true, if $\Sigma \models \sigma$, false otherwise

1. Initialization
TC=(V',E') := transitive closure of M(G)
for all $e \in E$ color(e) := black
for all $v \in V$ color(v) := black
for all $e' \in E'$ color(e') := blue
for all $v' \in V'$ color(v') := blue
for all $v_X \in X_1$ select $v\,(= v_X) \in V$ and set color(v) := green
for all $e_X \in X_2 \cap E$ select $e\,(= e_X) \in E$ and set color(e) := green
for all $v_X \in X_1$ select $v'\,(= v_X) \in V'$ and set color(v') := green
for all $e_X \in X_2$ select $e'\,(= e_X) \in E'$ and set color(e') := green
for all $v_Y \in Y_1$ ($v_Y \notin X_1$) select $v\,(= v_Y) \in V$ and set color(v) := yellow
for all $e_Y \in Y_2 \cap E(e_Y \notin X_2)$ select $e\,(= e_Y) \in E$ and set color(e) := yellow
for all $v_Y \in Y_1$ ($v_Y \notin X_1$) select $v'\,(= v_Y) \in V'$ and set color(v') := red
for all $e_Y \in Y_2(e_Y \notin X_2)$ select $e'\,(= e_Y) \in E'$ and set color(e') := red
2. $FDSET := \Sigma$;
3. greene := X (means, that $greene_1 := X_1$ and $greene_2 := X_2$);
4. repeat until no more dependency applicable:
if $W = (W_1, W_2) \rightarrow Z = (Z_1, Z_2) \in \Sigma$ and $W \subseteq greene$ (means, that $W_1 \subseteq greene_1$ and $W_2 \subseteq greene_2$) then
i. $FDSET := FDSET - (W \rightarrow Z)$;
ii. greene := greene $\cup$ Z (means, that $greene_1 := greene_1 \cup Z_1$ and $greene_2 := greene_2 \cup Z_2$);
iii. for all $v_Z \in Z_1$ select $v\,(= v_Z) \in V$ and set color(v) := green
iv. for all $e_Z \in Z_2 \cap E$ select $e\,(= e_Z) \in E$ and set color(e) := green
v. for all $v_Z \in Z_1$ select $v'\,(= v_Z) \in V'$ and set color(v') := green
vi. for all $e_Z \in Z_2$ select $e'\,(= e_Z) \in E'$ and set color(e') := green

5. if count_nodes(V,yellow) = count_nodes(V',red) = 0 and count_edges(E,yellow) = count_edges(E',red) = 0 then output true else output false.

Lemma 1. *The result of Algorithm 1 does not depend on the selection method used for Σ and $X \rightarrow Y$, respectively.*

Proof. The selection methods (after Definitions 1 - 2 and 3 - 4) affect the generation of subsequences in dual sentences based upon edges and nodes in the transitive closure of M(G). Algorithm 1 deals with edges and nodes, it does not trouble with the dual subsequences behind them, and because of Remark 2, all participants use the same selection methods, the steps of Algorithm 1 will not be influenced by the selection methods.

Proposition 1 (Functional Dependency Implication). *Let L(G) be regular language and let Σ be a set of FDs and let $X \rightarrow Y$ be an FD over L(G), then*

oarieli@mta.ac.il

$\Sigma \models X \to Y$ if and only if when the Algorithm 1 for input M(G), Σ and $X \to Y$ returns true. The implication does not depend on the selection method used for Σ and $X \to Y$, respectively.

Proof. Step 1. sets up an instance that hurts the dependency $X \to Y$ (the color green means that the two "tuples" have identical values on the attributes in X, but the colors yellow and red sign different values on the attributes of Y). The cycle 4 applies the dependencies from Σ so, that they force the instance to satisfy the from FDSET selected dependencies. In each step if for a $W \to Z$ dependency from the current FDSET $W \subseteq greene$ will not be fulfilled then we need not take it into account for the current step in the cycle, because the current state of the instance does satisfy this dependency (the nodes of W out of *greene* are either yellow or black in M(G) and they are either red or blue on the transitive closure TC, that is, the left side of the implication behind $W \to Z$ will not be satisfied, that means that the implication itself - denoted by the $W \to Z$ dependency - will be satisfied). When step 5 returns true, it means, that satisfying all of the dependencies in Σ causes satisfying of $X \to Y$, that is, the implication follows. When step 5 returns false, it means, that there is an instance, that satisfies all of the dependencies in Σ but it does not satisfy $X \to Y$, that is, the implication $\Sigma \models X \to Y$ will not be fulfilled.

We prove by contradiction that the result of the algorithm dos not depend on the sequence of the applied dependencies, that is, the resulting *greene* graph is unique for a given context of M(G), Σ and $X \to Y$. Let us assume, that two processing of the Algorithm 1, say A and B resulted $greene_A \neq greene_B$ and let $W^{(0)} \to Z^{(0)} \in \Sigma$ so, that $W^{(0)} \subseteq greene_A$ but $W^{(0)} \not\subseteq greene_B$. Then there is a $W^{(1)} \to Z^{(1)} \in \Sigma$ so, that it had to be applied in process A prior to $W^{(0)} \to Z^{(0)}$ but it had not been applied in process B at all. It follows that we come to a dependency $W^{(n)} \to Z^{(n)} \in \Sigma$ so, that $W^{(n)} \subseteq X$ and $W^{(n)} \not\subseteq greene_B$, a contradiction.

The last sentence follows from Lemma 1.

Remark 3 (Reasoning about Regular Functional Dependencies). The Algorithm 1 runs in quadratic time measured in the number of extended attributes appearing in Σ and $\sigma : X \to Y$ respectively. In the general case there seems no chance for a finite axiomatization: Arenas and Libkin have proven in [3] (based upon a proposition about k-ary axiomatization, proved by Abiteboul et al.in [1]) that the implication problem for XML functional dependencies is not finitely axiomatizable when the relevant part of the schema definition contains disjunctions and recurring components. This statement applies for regular functional dependencies too.

3.2 Regular FD on XML Schema Languages

We apply our definitions of functional dependency for XML schema languages. As presented in Section 2 elements of XML documents are sentences of regular languages and the corresponding dual language is the language generated by the

oarieli@mta.ac.il

element declaring regular expressions of DTD or XML Schema. We can associate a regular grammar to an XML document as follows:

Definition 7 (XML Element Grammar). *A regular grammar G(N,T,S,P) is associated with an XML document D, when there is a mapping A from nodes in D those element name is the non-terminal symbol S into the regular language L(G), so that:*

for each such node e and its child nodes $e_1, e_2, \ldots, e_m$*, there exists a chain of production rules*
$\{S \Rightarrow \langle N^{e_1}, N^{e_1}! \Rightarrow a_1 N^{e_2}, \ldots, N^{e_{m-1}} \Rightarrow a_{m-1} N^{e_m}, N^{e_m} \Rightarrow a_m \mathcal{ENP}, \mathcal{ENP} \Rightarrow \rangle\}$
in G so, that the terminal symbols (labels) of e_i *are* a_i*, and the array of non-terminal symbols* $N^{e_1}, N^{e_2}, \ldots, N^{e_m}$ *(associated to* $e_1, e_2, \ldots, e_m$*) matches the array of element names* $ele(e_1), \ldots, ele(e_m)$*. We say, that*
$A(e) = \{a_1, \ldots, a_m\}$

We consider examples with DTD element description and XML Schema complex type.

Regular FD on DTD Element Descriptions. Let us see an example XML document (Fig. 4), the corresponding DTD contains the following declaration:

```
<!ELEMENT course (Cid,(Stid,Stn)+)>
```

Let G=(N,T,S,P) be a regular language, with

$N = \{course, Cid, Stid, Stn, \mathcal{ENP}\}$
$T = \{\langle, \rangle, 10, 111, "John", \ldots\}$
$S = \{course\}$
P=$\{course \Rightarrow \langle Cid, Cid \Rightarrow 10\, Stid, Stid \Rightarrow 111\, Stn, Stn \Rightarrow "John"\ Stid\}$;
$\{Stid \Rightarrow 112\, Stn, Stn \Rightarrow "Mary", \ldots, \mathcal{ENP} \Rightarrow \rangle\}$;

The association A of the XML document on Fig. 4 against the grammar G gives e.g. $A(v_3) = \{20, 111, "John", 211, "Abe"\}$. It maps the data of the *course* nodes in our XML document to a subset in L(G). Using this association we can define the following key dependency:

$(\{Cid\}, \{\}) \rightarrow (\{\}, \{Stid, Stn\})$

There is another dependency, relating relevant student data (ID and name) coupled to a course and building pairs:

$(\{\}, \{Stid\}) \rightarrow (\{\}, \{Stn\})$

oarieli@mta.ac.il

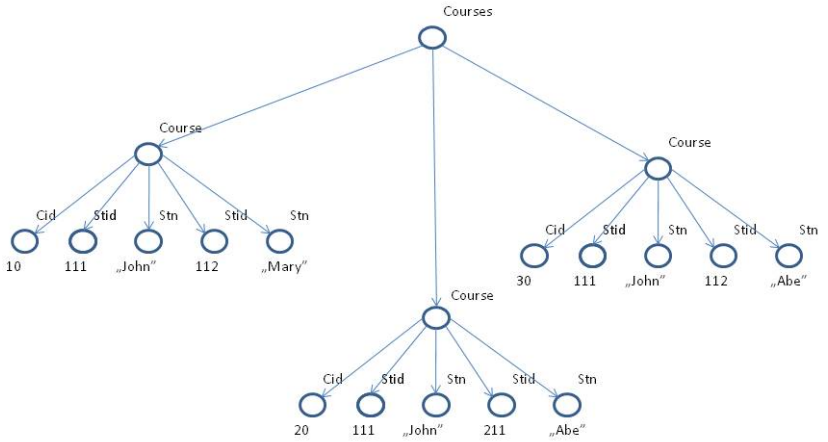


Fig. 4. Example XML data document (courses.xml)

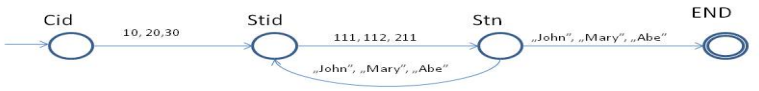


Fig. 5. Example FSA graph for a recursive element(Fig. 4)

using the association above and the definitions in Section 3.1. But neither of the Definitions 1 - 2 would offer a selection method to specify the desired constraint (Stid is the key for the next sibling Stn), so we need the passing Definitions 8-9, to deal with this case.

Definition 8 (Indexed Selection). *Let G=(N,T,S,P) be a regular grammar. Let L(G) be the corresponding regular language and let M(G) be the graph representation of the finite state automaton for L(G) and let w be a dual sentence over M(G). Let* $\omega = \{\omega_1, \ldots, \omega_{n+1}\}$ *be a path (circle) in w so, that the edge* $(\omega_{n+1}, \omega_1) \in M\,(G)$. *Let* $Y = \{y_1, \ldots, y_m\} \subseteq \omega, m \leq n-1$ *be a set of nodes to be selected. Let* $i \geq 1$ *be an index value. The nodes in Y will be selected for the index value i, denoted by* $w\left[Y^{(i)}\right]$ *while visiting them during the i-th closing of the circle* ω.

Definition 9 (Regular Functional Dependency Indexed). *Let L(G) be regular language and let M(G) be the graph representation of the finite state automaton for L(G) and let w be a dual sentence over M(G). Let X and Y be assignments in M(G), with selection specified after the Definition 8. An indexed*

oarieli@mta.ac.il

regular functional dependency (FD_{ind}) over M(G) is an expression of the form $X \xrightarrow{ind} Y$. *The* $R \subseteq L(G)$ *(finite) database instance over L(G) satisfies the* $X \xrightarrow{ind} Y$ *functional dependency (denoted by* $R \models X \xrightarrow{ind} Y$*), if for any two index values* $i, j \geq 1$, *and any two* $t_1, t_2 \in R$ *tuples (with type w)* $t_1\left[X^{(i)}\right] = t_2\left[X^{(j)}\right]$ *can be fulfilled only then, when* $t_1\left[Y^{(i)}\right] = t_2\left[Y^{(j)}\right]$ *comes true too.*

Regular FD on XML Schema Complex Type. The W3C specification [13] of XML Schema defines some integrity constraints (min and max occurs clauses,identity constraints which include uniqueness, key and foreign key definitions). Our definitions, applied for the XSD complex type, that is, to define an FD on the regular language generated by the regular expression (of element names of the children) specifying the given complex type could be put into the hierarchy of the predefined identity constraints. Our definitions can manage the cardinality constraints *min/maxOccurs*, the *sequence* and *choice* indicators (but the handling of the indicator *all* would be clumsy, we should take in fact *all* paths in account). Our current model does not allow the use of *XSD Attributes* and *XSD Mixed Type* at the moment.

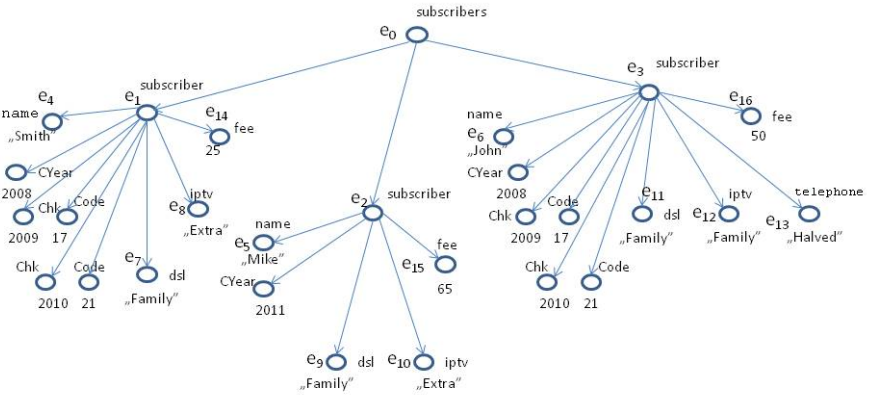


Fig. 6. Example XML document (t_com.xml)

Example 6. The XML document on Fig. 6 shows the subscribers data of a public telephone network. Figure 7 presents a fragment from the related XSD, the complex type definition "t_subscriber" for the element *subscriber*. This definition principally complies with the following regular expression:

$$E = \left(name, CYear, (Chk, Code)^*, dsl, telephone^?, iptv^?, fee\right)$$

oarieli@mta.ac.il

```
<xs:complexType name="t_subscriber">
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
    <xs:element name="name" type="xs:NCName"/>
    <xs:element name="CYear" type="xs:integer"/>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Chk" type="xs:integer"/>
        <xs:element name="Code" type="xs:integer"/>
        </xs:sequence>
    <xs:element name="dsl" type="xs:NCName"/>
    <xs:element minOccurs="0" name="iptv" type="xs:NCName"/>
    <xs:element minOccurs="0" name="telephone" type="xs:NCName"/>
    <xs:element name="fee" type="xs:integer"/>
  </xs:sequence>
</xs:complexType>
```

Fig. 7. Complex type definition for the element *subscriber* of Fig. 6

Let G=(N,T,S,P) be a regular language, with

$N = \{subscriber, name, CYear, Cid, Chk, Code, dsl, telephone, iptv, fee, \mathcal{END}\}$
$T = \{\langle, \rangle, "Smith", 2008, 17, 2009, \ldots\}$
$S = \{subscriber, \}$
P=$\{subscriber \Rightarrow \langle name, name \Rightarrow "Smith" \; CYear, CYear \Rightarrow 2008 \, Cid \ldots, \mathcal{END} \Rightarrow \rangle\}$;

The association A of the XML document on Fig. 6 with the grammar G gives e.g.

$$I\,(e_2) = \{"Mike", 2011, "Family", "Extra", 65\}$$

It maps the data of the type *t_subscriber* nodes in our XML document to a subset in L(G).

The element "CYear" specifies the beginning year of the connection for the given subscriber. From this year on the connection will be yearly checked: element "Chk" contains the year as the line has been checked, the element "Code" fixes a specific code that depends upon "CYear" and the year of checking. We have the following FD (with any selection type) using the association above:

$$(\{CYear\}, \{\}) \rightarrow (\{\}, \{Chk, Code\})$$

We have for the subscribers "Smith" and "John" (2008) as left side, and the same right side (2009 17 2010 21), so the instance satisfies the dependency.

Example 7. The XSD in Fig. 7 describes the data types and their relations for the subscribers of a public telephone network. Figure 6 shows an instance of the XSD in Fig. 7. The services consumed by a given subscriber are not always complete, the subscribing of telephone or iptv service is optional: it is allowed, to use dsl without any telephone connection, to use iptv or do not use it at all. The fee to be paid monthly depends on the consumed services (e.g. Telephone:

oarieli@mta.ac.il

Basic=5,Favorite=10,Halved=15 etc.), that is, when the consumed services are the same for two subscribers, then their monthly fees should also be equal. A functional dependency holds between the services consumed by and the total fee payed by a given subscriber:

FD: $(\{dsl, telephone, iptv\}, \{\}) \rightarrow (\{fee\}, \{\}))$

This functional dependency selects from the 4 possible dual sentences different subsequences, so it can handle incomplete data perfectly:

$w_1 = \{\ldots dsl\ fee\};\ w_1[X] = \{dsl\}, w_1[Y] = \{fee\}$
$w_2 = \{\ldots dsl\ telephone\ fee\};\ w_2[X] = \{dsl\ telephone\}, w_2[Y] = \{fee\}$
$w_3 = \{\ldots dsl\ iptv\ fee\};\ w_3[X] = \{dsl\ iptv\}, w_3[Y] = \{fee\}$
$w_4 = \{\ldots dsl\ telephone\ iptv\ fee\};\ w_4[X] = \{dsl\ telephone\ iptv\}, w_4[Y] = \{fee\}$

We have for the subscribers "Smith" and "Mike" ("Family","Extra","") as left side, and different right sides (25 and 65 resp.), so the instance does not satisfy the dependency.

Fig. 8. Example FSA graph for an XML element type with incomplete data

4 Comparing Regular FD with XFDs

We have shown in the last section that the Regular FD can be well interpreted in the XML world. There are a lot of valuable XML functional dependency (XFD) definitions published: Wang in 2005 [15] compared many different functional dependency definitions for XML and proposed a new definition of XFD which unifies and generalizes the surveyed XFDs: all these XFD definitions base upon path expressions created from DTDs or XML Schema definitions. We compare here the tree tuple concept of Arenas and Libkin [3] with our regular FD using an example.

Example 8. The DTD in Fig. 9 describes the same data set as the one discussed in Example 6, a passing XML document is shown on Fig. 6.

oarieli@mta.ac.il

```
<!DOCTYPE subscribers [
<!ELEMENT subscribers (subscriber*)>
<!ELEMENT subscriber (name, CYear, (Chk, Code)*, dsl, iptv?, telephone?, fee)>
<!ELEMENT name (#PCDATA)>
...
<!ELEMENT fee (#PCDATA)>
]>
```

Fig. 9. An example DTD for the XML tree of Fig. 6

We omit the definitions of tree tuples, the subsumption among XML trees, the conformance and compability of an XML tree to a DTD, and the exact definitions of $trees_D, tuples_D$ (We refer to the article of Arenas and Libkin [3]). $tuples_D$ is the set of tree tuples that construct an XML tree T and connect it to the DTD D that describes the given XML tree T.

Example 9. Example 8 suggests an integrity constraint for the data on Fig. 6: the fee payed by a subscriber depends upon the services consumed by the given subscriber. This constraint can be described using the concept of tree tuples with the following paths:

S_1:subscribers.subscriber.dsl,
subscribers.subscriber.iptv,
subscribers.subscriber.telephone $\rightarrow$ S_2: subscribers.subscriber.fee

This FD can detect the violations when we make the empty ($\perp$) values equal: (denote p: subscriber.telephone) $t_1.p = t_2.p = \perp$ and follows that $t_1.S_1 = t_2.S_1$ but $25 = t_1.S_2 \neq t_2.S_2 = 65$.

The passing regular FD can be constructed as

FD: $(\{dsl, iptv, telephone\}, \{\}) \rightarrow (\{fee\}, \{\}))$

Example 10. Let us slightly modify the DTD above with

```
<!ELEMENT subscriber (name, CYear, (Chk, Code)*, services, fee)>
<!ELEMENT services (dsl, iptv?, telephone?)>
```

then we get the XFD for tree tuples as

S_1:subscribers.subscriber.services.dsl,
subscribers.subscriber.services.iptv,
subscribers.subscriber.services.telephone $\rightarrow$ S_2: subscribers.subscriber.fee

oarieli@mta.ac.il

Let t_1 be the tree tuple fitting on the vertex e_1 (the subtree of Smith), then:
t_1(subscribers)=e_0
t_1(subscribers.subscriber)=e_1
t_1(subscribers.subscriber.name)=e_4
t_1(subscribers.subscriber.name.S)="Smith"
t_1(subscribers.subscriber.fee)=e_{14}
t_1(subscribers.subscriber.fee.S)="25"
t_1(subscribers.subscriber.dsl)=e_7
t_1(subscribers.subscriber.dsl.S)="Family"
t_1(subscribers.subscriber.iptv)=e_8
t_1(subscribers.subscriber.iptv.S)="Extra"
t_1(subscribers.subscriber.telephone)=$\perp$
and t_2 be the tree tuple fitting on the vertex e_2 (the subtree of Mike), then:
t_2(subscribers)=e_0
t_2(subscribers.subscriber)=e_2
t_2(subscribers.subscriber.name)=e_5
t_2(subscribers.subscriber.name.S)="Mike"
t_2(subscribers.subscriber.fee)=e_{15}
t_2(subscribers.subscriber.fee.S)="65"
t_2(subscribers.subscriber.dsl)=e_9
t_2(subscribers.subscriber.dsl.S)="Family"
t_2(subscribers.subscriber.iptv)=e_{10}
t_2(subscribers.subscriber.iptv.S)="Extra"
t_2(subscribers.subscriber.telephone)=$\perp$

Fig. 10. Two example tree tuples for the DTD and XML tree of Fig. 6

The corresponding regular FD appears as

FD: $(\{services\}, \{\}) \rightarrow (\{fee\}, \{\}))$

It can be seen, that tree tuples deal with the exact selection of each participant element for the FD, the regular FD, however, focuses on the dependency connection between complex valued data.

5 Conclusion and Future Works

Functional dependency is a constraint that can help to detect corrupt data in a data set. But functional dependency is also a tool for the database (document) administrator supporting him or her to create a consistent data structure. Functional dependency for relational data model is easy to understand and it can be applied easily for database design. The known FD concepts for other data models (e.g. XML and semantic data models) are very intricate. We wanted to propose a general functional dependency concept that fits for the relational and other (e.g. XML) data models as well.

Our model can deal with the XML schema language DTD well, but it can not manage some properties of W3C XML Schema (*XSD Mixed Type*,*XSD Element*

oarieli@mta.ac.il

Substitution, *XSD Restriction* etc.). We would like to extend our model to deal with them too.

Another future research direction would be to use the concept of regular grammar for describing the complex value model (tuple and set constructor) and for defining dependencies on this model. We can specify the structure of the complex value with a regular expression, so that each constructor would have a distinct name. This differs from the traditional concept of complex values used by Abiteboul et al. [1], but no more than writing type names in state of constructor symbols. We can also say, that the notation will have an XML character. The name of the constructor corresponds to an XML tag.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Amano, S., Libkin, L., Murlak, F.: XML schema mappings. In: Proceedings of the Twenty-Eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Providence, Rhode Island, USA, June 29-July 01, pp. 33–42 (2009)
3. Arenas, M., Libkin, L.: A normal form for XML documents. ACM TODS 29, 195–232 (2004)
4. Buneman, P., Davidson, S.B., Fan, W., Hara, C.S., Tan, W.C.: Keys for XML. Computer Networks 39(5), 473–487 (2002)
5. Chen, Y., Davidson, S.B., Zheng, Y.: Constraint Preserving XML Storage in Relations. In: Proceedings of the International Workshop on the Web and Databases (WEBDB), pp. 7–12 (2002)
6. Lee, D., Mani, M., Murata, M.: Reasoning about XML Schema Languages using Formal Language Theory. Technical Report, IBM Almaden Research Center, RJ# 10197, Log# 95071 (2000), http://www.cs.ucla.edu/~dongwon/paper
7. Davidson, S., Fan, W., Hara, C.: Propagating XML constraints to relations. Journal of Computer and System Sciences 73(3), 316–361 (2007)
8. Hartmann, S., Köhler, H., Trinh, T.: On the Existence of Armstrong Data Trees for XML Functional Dependencies. In: Link, S., Prade, H. (eds.) FoIKS 2010. LNCS, vol. 5956, pp. 94–113. Springer, Heidelberg (2010)
9. Hartmann, S., Link, S.: Efficient reasoning about a robust XML key fragment. ACM Trans. Database Syst. 34(2) (2009)
10. Hartmann, S., Link, S., Trinh, T.: Solving the Implication Problem for XML Functional Dependencies with Properties. In: Dawar, A., de Queiroz, R. (eds.) WoLLIC 2010. LNCS, vol. 6188, pp. 161–175. Springer, Heidelberg (2010)
11. Lv, T., Yan, P.: Mapping Relational Schemas to XML DTDs with Constraints. In: Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences, pp. 528–533 (2006)
12. Murata, M., Lee, D., Mani, M., Kawaguchi, K.: Taxonomy of XML schema languages using formal language theory. ACM Trans. Internet Technol. 5(4), 660–704 (2005)

oarieli@mta.ac.il

13. Sperberg-McQueen, C.M., Thompson, H.: XML Schema. Technical report, World Wide Web Consortium (2005), http://www.w3.org/XML/Schema
14. Vincent, M.W., Liu, J., Liu, C.: Strong functional dependencies and their application. to normal forms in XML. ACM ToDS 29, 445–462 (2004)
15. Wang, J.: A Comparative Study of Functional Dependencies for XML. In: Zhang, Y., Tanaka, K., Yu, J.X., Wang, S., Li, M. (eds.) APWeb 2005. LNCS, vol. 3399, pp. 308–319. Springer, Heidelberg (2005)

oarieli@mta.ac.il

Author Index

Arieli, Ofer 34

Beierle, Christoph 224
Benczúr, András 384
Benczúr, András A. 244
Biskup, Joachim 51
Bock, Jürgen 71

Caroprese, Luciano 91
Cohen, Andrea 107

Doder, Dragan 264

Eiter, Thomas 1

Faruqui, Rokan Uddin 312
Fletcher, George H.L. 124

García, Alejandro J. 107
Garnaud, Eve 144
Guzzo, Antonella 352
Gyssens, Marc 124

Hartmann, Sven 164
Hegner, Stephen J. 184

Katona, Gyula O.H. 203
Kifer, Michael 332
Köhler, Henning 164
Krämer, Annika 224
Krennwallner, Thomas 1

Leinders, Dirk 124
Link, Sebastian 164

Maabout, Sofian 144
MacCaull, Wendy 312
Molnár, András J. 244
Mosbah, Mohamed 144
Mutter, Michael 71

Ognjanović, Zoran 264

Perović, Aleksandar 264
Pivert, Olivier 278
Prade, Henri 278

Rajagopalan, Krishnan 298
Rakib, Abdur 312
Rezk, Martín 332
Rudolph, Sebastian 71

Saccà, Domenico 352
Schewe, Klaus-Dieter 370
Schneider, Patrik 1
Schwentick, Thomas 23
Serra, Edoardo 352
Sidló, Csaba István 244
Simari, Guillermo R. 107
Srinivasan, Venkatesh 298
Szabó, Gyula I. 384

Tadros, Cornelia 51
Thalheim, Bernhard 164
Thomo, Alex 298
Tichler, Krisztián 203

Van den Bussche, Jan 124
Van Gucht, Dirk 124
Vansummeren, Stijn 124

Wang, Qing 370
Wu, Yuqing 124

Xiao, Guohui 1

Zamansky, Anna 34
Zumpano, Ester 91

oarieli@mta.ac.il